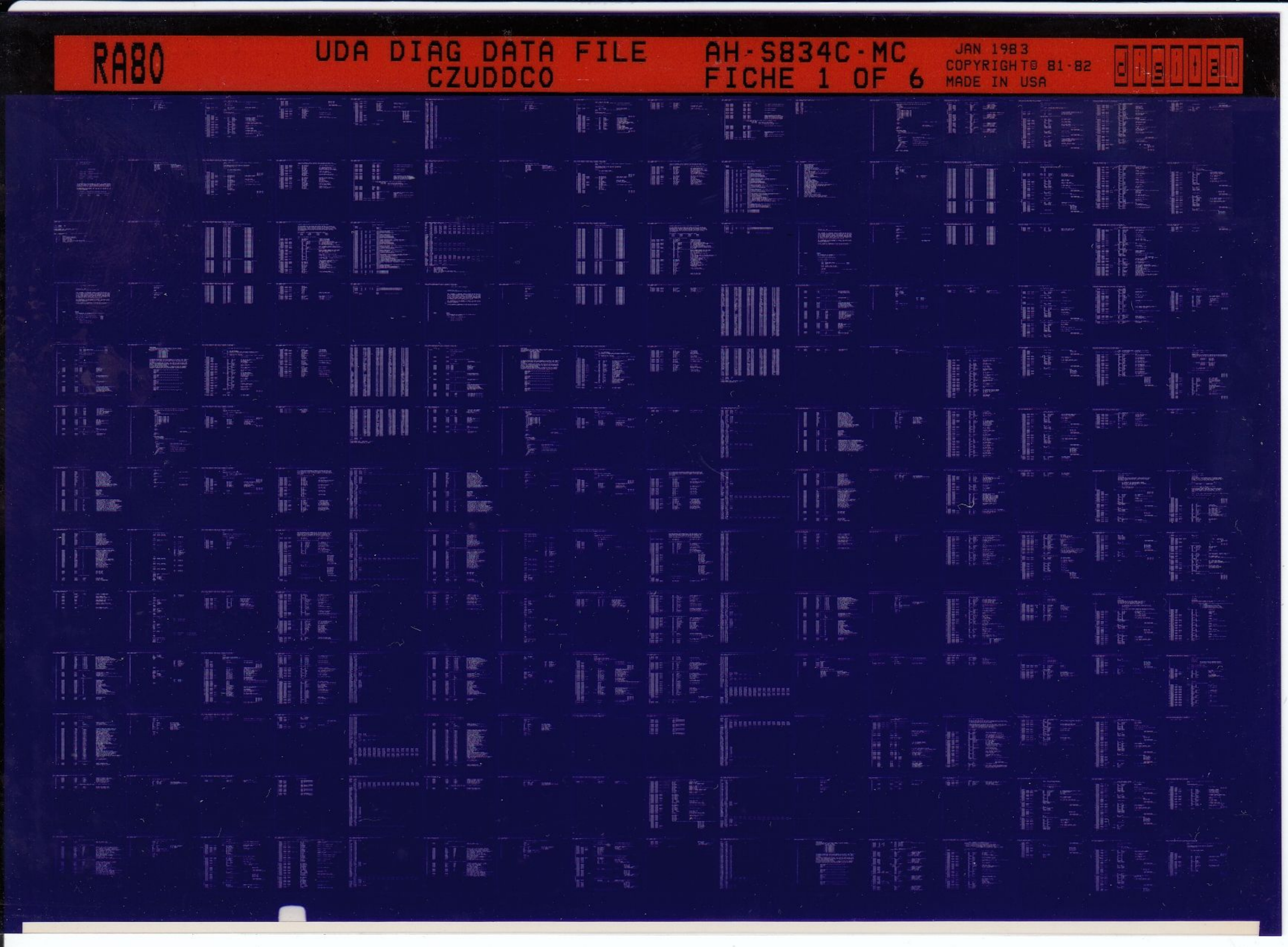
RA80

UDA DIAG DATA FILE
CZUDDCO

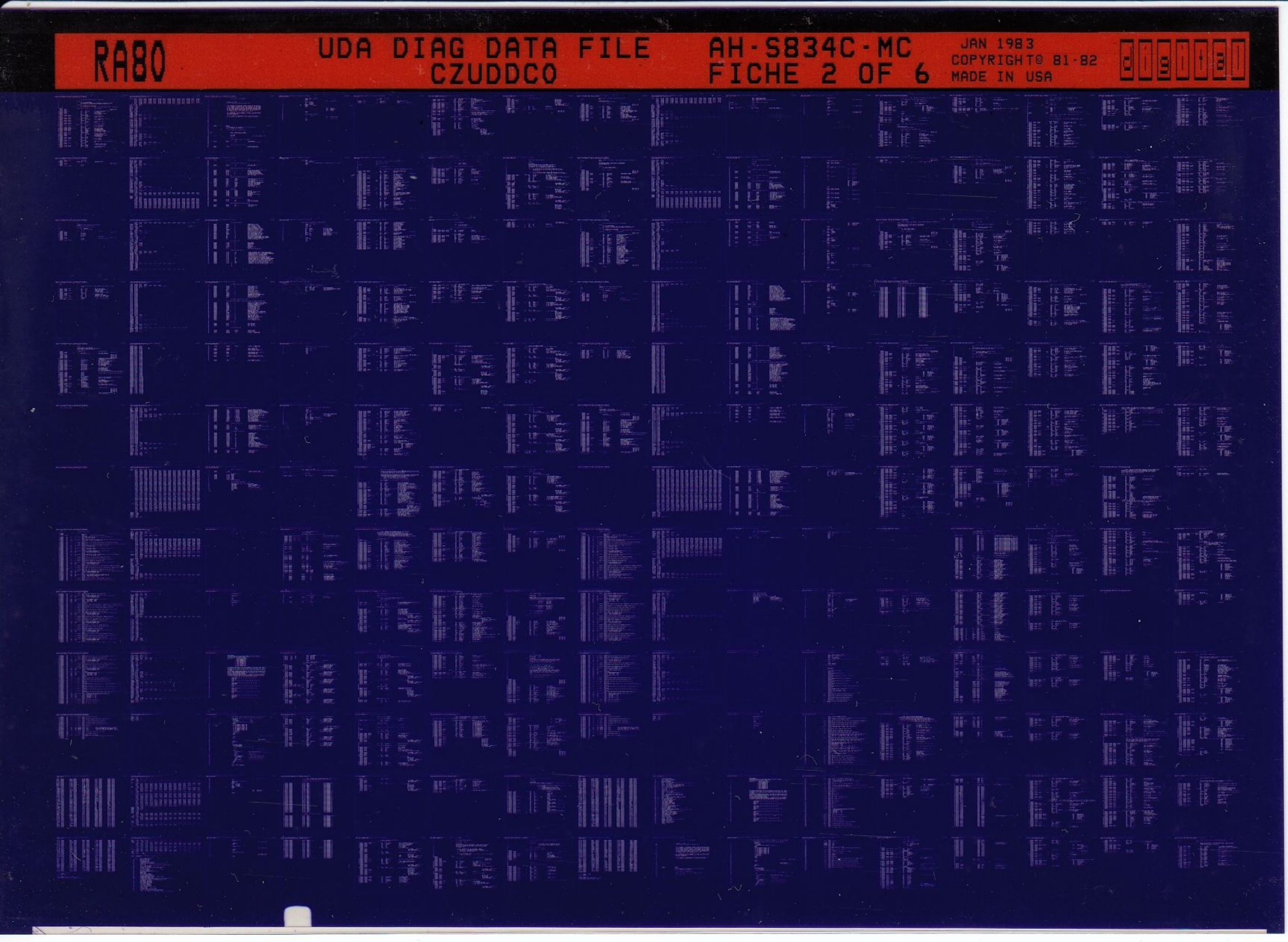
AH-S834C-MC
FICHE 1 OF 6



RA80

UDA DIAG DATA FILE
CZUDDCO

AH-S834C-MC
FICHE 2 OF 6



RA80 UDA DIAG DATA FILE AH-S834C-MC JAN 1983
CZUDDC0 FICHE 3 OF 6
MADE IN USA

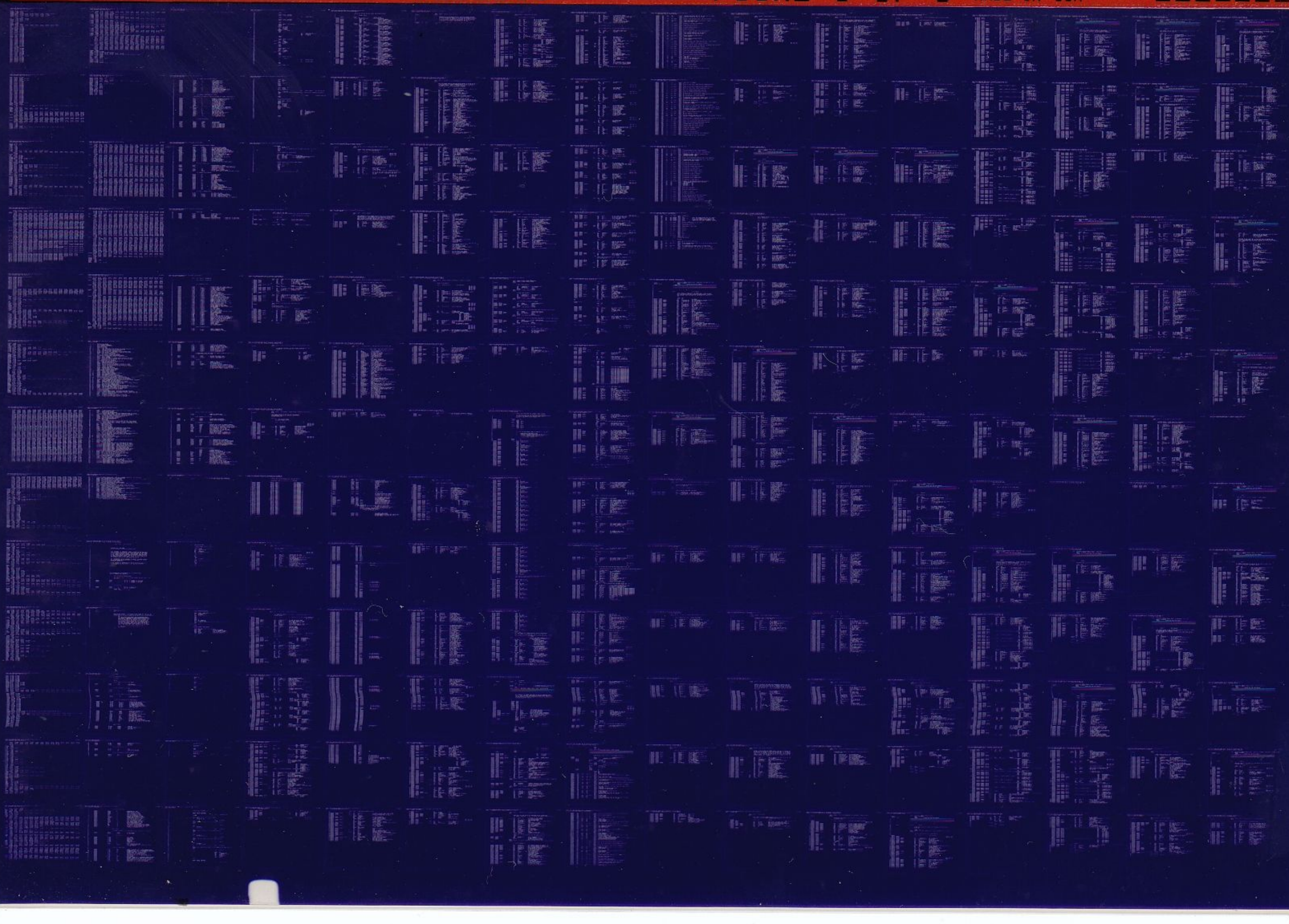
RA80

UDA DIAG DATA FILE
CZUDDC0

AH-S834C-MC
FICHE 4 OF 6

d i g i t a l

RA80 UDA DIAG DATA FILE AH-S834C-MC JAN 1983
CZUDDC0 FICHE 5 OF 6
MADE IN USA



RA80

UDA DIAG DATA FILE
CZUDDCO

AH-S834C-MC
FICHE 6 OF 6

CZUDDCO UDA DIAG DATA FILE MACRO V04.00 23-AUG-82 17:24:32 SEQ 0001
TABLE OF CONTENTS

```
                        .TITLE  CZUDDCO UDA DIAG DATA FILE
                        .SBTTL  USER DOCUMENTATION
                        .REM ~
```

IDENTIFICATION

PRODUCT CODE: AC-S833C-MC

PRODUCT NAME: CZUDDCO UDA DIAG DATA FILE

PRODUCT DATE: 27-AUG-82

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: MATT TEDONE


```
                        ~
000001                  .END
```

```
SYMBOL TABLE


. ABS.  000000     000
        000000     001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  8192 WORDS  ( 32 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  72 PAGES
,B:ZUDDCO=B:ZUDDCO.TTL


UDAT1   UNIBUS ADDRESSING       DMACR X04.01 23-AUG-82 13:10:39
TABLE OF CONTENTS

    2-   5      MCALLS                                          ;UDA52
    2-  22      UDA DM PROGRAM PARAMETERS
    6-   1      TEST 4 SPECIFIC INFORMATION
    8-   1      MACRO DEFINITIONS
   19-   1      START OF TEST CODE
   19-  31      RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
   20-   1      HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
   29-   1      FREE MEMORY CHECK
```

```
                                .TITLE  UDAT1   UNIBUS ADDRESSING
                        ;
                        ;
                        ; COPYRIGHT (C)  1981
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
000000                  ;UDA50=0
000001                  ;UDA52=1
                        ;THIS PROGRAM SHALL BE ASSEMBLED WITH THE PROGRAM DMACRO
                        ;USING A COMMAND LINE SIMILAR TO:
                        ;
                        ;  UDAT1.OBJ,UDAT1/C=[30,30]DMAC52/M,UDAT1      ;UDA52
                        ;                                               ;UDA52
                        ;THEN LINK USING A COMMAND LINE SIMILAR TO:     ;UDA52
                        ;                                               ;UDA52
                        ;  UDAT1.BIN/L=UDAT1                            ;UDA52
                        ;
                        ;       VERSION 1.0 FOR RELEASE
                        ;
                        ;       VERSION 2.0 FOR NEXT UDA
                        ;
```

```
                                TEST4   =       0                       ; THIS IS NOT TEST4
                                        .SBTTL MCALLS                                   ;UDA52

                                        .MCALL  DMCODE,DMEND,DMOVLY                     ;UDA52
                                        .MCALL  JMP,BR,BEQ,CALL,BPL,BCC,BNE,BMI,RETURN  ;UDA52
                                        .MCALL  DMODT                                   ;UDA52

000000                                  DMCODE  UDADM1,0,1364,3,0,1000                  ;UDA52

                                .SBTTL  UDA DM PROGRAM PARAMETERS

                                        .LIST MEB
                                ;
                                ;       EQUATES
                                ;
000001                          BIT00 = 1
000002                          BIT01 = 2
                                ;
                                ;       HIGHEST USABLE LOCATION OF UDA MEMORY + 1
                                ;
037777                          HIMEM   =       37777
                                ;
                                ;       DUP PACKET VALUE
                                ;
010000                          DU.QUE = 10000                          ;QUESTION
020000                          DU.DFL = 20000                          ;DEFAULT QUESTION
030000                          DU.INF = 30000                          ;INFORMATION
040000                          DU.TER = 40000                          ;TERMINATOR
050000                          DU.FTL = 50000                          ;FATAL ERROR
060000                          DU.SPC = 60000                          ;SPECIAL
                                ;
                                ;       OFFSETS FOR FORMAT TRACK TABLE
                                ;
000000                          FT.BUF  =       0.                      ;BUFFER POINTER OFFSET
000001                          FT.HI   =       1.                      ;HI ORDER HEADER OFFSET
000002                          FT.LOW  =       2.                      ;LOW ORDER HEADER OFFSET
                                ;
                                ;
                                ;       OFFSETS FOR FORMAT TRACK BUFFER
                                ;
000000                          FB.DAT  =         0.                    ;FIRST DATA WORD OFFSET
000400                          FB.EDC  =       256.                    ;EDC WORD OFFSET
                                ;
                                ;
                                ;       OFFSETS FOR READ/WRITE I/O CHAIN TABLES
                                ;
000000                          RW.STAT =       0.                      ;STATUS AND NEXT BUFFER POINTER OFFSET
000001                          RW.BUF  =       1.                      ;POINTER TO DATA BUFFER
000002                          RW.LOW  =       2.                      ;HI ORDER EXPECTED HEADER
000003                          RW.HI   =       3.                      ;LOW ORDER EXPECTED HEADER
000004                          RW.CMD  =       4.                      ;SDI COMMAND AND HEAD ADDRESS
000005                          RW.SDI  =       5.                      ;DUMMY SDI CONTROL BLOCK POINTER
000006                          RW.ANG  =       6.                      ;THETA FROM INDEX
                                ;
                                ;       CONSTANTS FOR READ AND WRITE XFC'S
                                ;
140000                          WSTOP   =       140000                  ; LAST ENTRY IN CHAIN FOR WRITE
```

```
        040000          WCONT   =       40000           ; WRITE CONTINUE
        100000          RSTOP   =       100000          ; LAST ENTRY IN CHAIN FOR READ
        000000          RCONT   =       0               ; READ CONTINUE
        100000          FSTOP   =       100000          ; LAST ENTRY IN CHAIN FOR FORMAT
        122400          WREAL   =       122400          ; WRITE REAL TIME ECOMMAND
        013400          RREAL   =       13400           ; READ REAL TIME COMMAND
        010000          ECCFLG  =       10000           ; ECC ERROR IN BUFFER BIT
        100000          EOC     =       100000          ; END OF CHAIN
        040000          BUFFLG  =       40000           ; BUFFER FULL OR EMPTY FLAG
                        ;
                        ;
                        ;       HEADER CODES
                        ;
        000000          HD.LBN  =       000000          ;GOOD LBN
        060000          HD.RBN  =       060000          ;GOOD RBN, PERHAPS UNUSED
        030000          HD.REV  =       030000          ;REVECTORED LBN
        110000          HD.BAD  =       110000          ;BAD BLOCK
        050000          HD.PRV  =       050000          ;PRIMARY REVECTORED BLOCK
        120000          HD.XBN  =       120000          ;XBN BLOCK
        140000          HD.DBN  =       140000          ;DBN BLOCK

                        ;       OFFSETS FOR DATA BUFFERS

        000000          BF.DAT  =         0.            ;DATA
        000400          BF.EDC  =       256.            ;ERROR DETECTION CODE
        000401          BF.ECC  =       257.            ;LAST 17 ECC RESIDUES

                        ;       BUFFER AND READ/WRITE CHAIN LINK SIZES

        000401          WBUFLN  =       257.            ; WRITE BUFFER SIZE
        000415          RBUFLN  =       WBUFLN+12.      ; READ BUFFER SIZE
        000007          LINKLN  =       7.              ; LINK SIZE
```

```
                        ;       XFC DEFINITION EQUATES
                        ;
000000                  BREAK   =       0.              ;BREAKPOINT XFC CODE
000001                  FORMAT  =       1.              ;FORMAT TRACK XFC CODE
000002                  XREAD   =       2.              ;READ N SECTORS XFC CODE
000003                  XWRITE  =       3.              ;WRITE N SECTORS XFC CODE
000004                  SEND    =       4.              ;SEND SDI COMMAND XFC CODE
000005                  RCV     =       5               ;RECEIVE SDI MESSAGE XFC CODE
000006                  COMPARE =       6.              ;COMPARE DATA PATTERN TO BUFFER
000007                  STATUS  =       7.              ;RETURN DRIVE STATUS XFC CODE
000010                  ECHO    =       8.              ;ECHO DATA TO DRIVE XFC CODE
000011                  DINIT   =       9.              ;DRIVE INITIALIZE XFC CODE
000012                  WAITSI  =       10.             ;WAIT FOR SECTOR OR INDEX PULSE
000013                  UREAD   =       11.             ;READ UNIBUS MEMORY XFC CODE
000014                  UWRITE  =       12.             ;WRITE UNIBUS MEMORY XFC CODE
000015                  ECC     =       13.             ;DO ECC ON BUFFER XFC CODE
000016                  MRD     =       14.             ;SEND BUFFER TO MAINTENANCE READ COMMAND
000017                  MWR     =       15.             ;GET BUFFER FROM MAINTENANCE WRITE COMMAND
000020                  CVT     =       16.             ;CONVERT TO PHYSICAL ADDRESS XFC CODE
000021                  EXIT    =       17.             ;TERMINATE DM PROGRAM XFC CODE
                        ;
                        ;       GET STATUS OFFSETS
                        ;
000000                  ST.UNT  =       0.              ;UNIT NUMBER
000000                  ST.MSK  =       0.              ;SUBUNIT MASK
000001                  ST.STA  =       1.              ;STATUS BYTE
000001                  ST.MOD  =       1.              ;MODE BYTE
000002                  ST.ERR  =       2.              ;ERROR BYTE
000002                  ST.CON  =       2.              ;CONTROLLER BYTE
000002                  ST.C    =       2.              ;C BITS
000003                  ST.RTY  =       3.              ;RETRY COUNT/FAILURE CODE
                        ;
                        ;       STATUS BIT DEFINITIONS
                        ;
000200                  ST.OA   =       200             ; ONLINE TO ANOTHER (SET IF DRIVE UNAVAILABLE)
000100                  ST.RR   =       100             ; READJUSTMENT BIT (SET IF RECALIBRATION REQUIRED)
000040                  ST.DR   =       40              ; DIAGNOSTIC REQUEST (SET IF DIAGNOSTIC REQUESTED)
000020                  ST.SR   =       20              ; SPINDLE READY (SET IF SPINDLE READY)
000002                  ST.PS   =       2               ; PORT SWITCH (SET IF PORT SWITCH IN)
000001                  ST.RU   =       1               ; RUN/STOP SWITCH (SET IF RUN/STOP SWITCH IN)
000200                  ST.FE   =       200             ; FATAL ERROR (SET IF FATAL ERROR OCCURRED)
000100                  ST.RE   =       100             ; RETRIABLE ERROR (SET IF RETRIABLE ERROR OCCURRED)
000040                  ST.PE   =       40              ; PROTOCOL ERROR (SET IF PROTOCOL ERROR OCCURRED)
000020                  ST.DF   =       20              ; INITIALIZATION FAILURE (SET IF INIT FAILED)
000010                  ST.WE   =       10              ; WRITE ENABLE (SET IF WRT ATTEMPTED ON PROT DISK)
002000                  ST.FO   =       2000            ; FORMATTING (SET IF FORMATTING ENABLED)
001000                  ST.DB   =       1000            ; DIAGNOSTIC CYLS (SET IF DIAG CYL ACCESS ENABLED)
000400                  ST.S7   =       400             ; SECTOR SIZE (SET FOR 576 BYTE SECTORS)
```

```
                        ;       GET COMMON CHARACTERISTICS OFFSETS
                        ;
000000          SHRTTO  =       0.              ;SHORT TIMEOUT <3:0>
000000          SDIVER  =       0.              ;SDI VERSION <7:4>
000000          XFERRT  =       0.              ;TRANSFER RATE <15:0>
000001          LONGTO  =       1.              ;LONG TIMEOUT <3:0>
000001          RETS    =       1.              ;RETRIES <7:4>
000001          RCTCPS  =       1.              ;F/RCT COPIES <11:8>
000001          SS      =       1.              ;SECTOR SIZE <15:15>
000002          ERLEV   =       2.              ;ERROR RETRY LEVELS <7:0>
000002          ECCRSH  =       2.              ;ECC THRESHOLD <15:8>
000003          MICREV  =       3.              ;MICROCODE REVISION NUMBER <7:0>
000003          HRDREV  =       3.              ;HARDWARE REVISION NUMBER <15:8>
000004          DRVID   =       4.              ;UNIQUE DRIVE ID <47:0>
000007          DRTYPE  =       7.              ;DRIVE TYPE IDENTIFIER <7:0>
000007          REVS    =       7.              ;REVS/SECOND <15:8>
                        ;
                        ;       GET SUBUNIT CHARACTERISTICS OFFSETS
                        ;
                        ;THESE OFFSETS ARE CURRENTLY GIVEN AS FOLLOWING THE COMMON CHARACTERISTICS
                        ;
000013          SUB     =       11.             ;OFFSET TO PUT SUBUNIT AFTER COMMON;
000000          LBNCYL  =       0.              ;NUMBER OF CYLINDERS IN LBN AREA <31:0>
000001          HICYL   =       1.              ;HI ORDER CYLINDER BITS <15:12>
000002          GRPCYL  =       2.              ;GROUPS PER CYLINDER <7:0>
000002          HILBN   =       2.              ;HI STARTING LBN <11:8>
000002          HIXBN   =       2.              ;HI STARTING XBN <15:12>
000003          TRKGRP  =       3.              ;TRACKS PER GROUP <7:0>
000003          HIRBN   =       3.              ;HI STARTING RBN <11:8>
000003          HIDBN   =       3.              ;HI STARTING DBN <15:12>
000004          RBNTRK  =       4.              ;RBNS PER TRACK <6:0>
000004          RM      =       4.              ;REMOVABLE MEDIA <7:7> 1=REMOVEABLE
000005          DATPRE  =       5.              ;DATA PREAMBLE SIZE IN WORDS <7:0>
000005          HDRPRE  =       5.              ;HEADER PREAMBLE SIZE IN WORDS <15:8>
000006          MEDTYP  =       6.              ;MEDIA TYPE <31:0>
000010          FCTSIZ  =       8.              ;FCT COPY SIZE <15:0>
000011          LBNTRK  =       9.              ;LBNS PER TRACK <7:0>
000011          GRPOFF  =       9.              ;GROUP OFFSET (SECTORS) <15:8>
000012          LBNHST  =       10.             ;LBNS IN HOST AREA <31:0>
000014          RCTCSZ  =       12.             ;RCT COPY SIZE <15:0>
000021          XBNCYL  =       17.             ;CYLS IN XBN AREA <15:0>
000022          DBNCYL  =       18.             ;CYLS IN DBN AREA <15:8>
                        ;
                        ;       UNIT CODES
                        ;
000001          UNIT0   =       1.              ;UNIT ZERO CODE
000002          UNIT1   =       2.              ;UNIT ONE CODE
000004          UNIT2   =       4.              ;UNIT TWO CODE
000010          UNIT3   =       8.              ;UNIT THREE CODE
                        ;
                        ;       BIT MASK DEFINITIONS
                        ;
                        ;
177400          HIBYTE  =       177400          ;HIGH BYTE MASK
000377          LOBYTE  =       000377          ;LOW BYTE MASK
007777          HBHINB  =       7777            ;HI BYTE, HI NIBBLE MASK
170377          HBLONB  =       170377          ;HI BYTE, LO NIBBLE MASK
```

```
      58          177417                LBHINB  =       177417          ;LO BYTE, HI NIBBLE MASK
      59          177760                LBLONB  =       177760          ;LO BYTE, LO NIBBLE MASK
      60                                ;
      61          000001                TIMEOUT =       1.              ;DRIVE TIMEOUT CODE
      62          000002                HEADER  =       2.              ;HEADER COMPARE FAILURE CODE
      63          000004                REVECT  =       4.              ;REVECTOR NEEDED CODE
      64                                ;
      65          000002                WRONG   =       2.              ;FIRST WORD NOT START FRAME CODE
      66          000004                FRAME   =       4.              ;FRAMING ERROR CODE
      67          000010                CHECK   =       8.              ;CHECKSUM ERROR CODE
      68                                ;
      69          000001                TOOBIG  =       1.              ;NUMBER OF WORDS EXCEEDS 7064
      70          000002                LOW     =       2.              ;DM BUFFER ADDRESS IS LESS THAN 714
      71                                ;
      72                                ;       BIT MASK
      73                                ;
      74          000001                LARGE   =       1.              ;BLOCK NUMBER TOO LARGE
      75          000002                OVERFL  =       2.              ;SECTOR NUMBER LARGER THAN 16 BITS
```

```
                                ;MAINTANENCE READ/WRITE REQUEST NUMBERS
                                ;
060000                          T1MSIZ  =       0.+DU.SPC               ;GET FREE MEMORY PARAMETERS
060001                          T2DLL   =       1.+DU.SPC               ;DOWNLINE LOAD DRIVE DIAGNOSTIC
060002                          T2CMD   =       2.+DU.SPC               ;MANUAL INTERVENTION TEST 2 PROTOCOL
060003                          T4MPRM  =       3.+DU.SPC               ;GET MASTER PARAMETERS FROM SW QUESTIONS
060004                          T4UPRM  =       4.+DU.SPC               ;GET UNIT PARAMETERS FROM HW QUESTIONS
060005                          T4BB1   =       5.+DU.SPC               ;GET BAD BLOCKS (1 THRU 14)
060006                          T4BB2   =       6.+DU.SPC               ;GET REST OF BAD BLOCKS (15 AND 16)
060007                          T4SOFT  =       7.+DU.SPC               ;ADD TO SOFT ERROR AND ECC COUNT
060010                          T4SEEK  =       8.+DU.SPC               ;ADD 1 TO SEEK COUNT
060011                          T4MXFR  =       9.+DU.SPC               ;ADD TO MEGABITS READ AND WRITTEN
060012                          UTOTST  =       10.+DU.SPC              ;GET UNITS TO TEST
060013                          ERRMES  =       11.+DU.SPC              ;PRINT ERROR MESSAGE
060014                          ERRMC   =       12.+DU.SPC              ;TEST 4 ERROR REPORTING
060015                          MESSAG  =       13.+DU.SPC              ;INFORMATION MESSAGE
060016                          DONE    =       14.+DU.SPC              ;MARK DM PROGRAM AS NO LONGER RUNNING
                                ;
                                ;       OTHER BIT DEFINITIIONS
                                ;
000001                          RCVRDY  =       1                       ; RECIEVER READY 1 = READY
000002                          ATTN    =       2                       ; ATTENTION BIT FOR RETURN DRIVE SIGNALS XFC
000004                          RCVERR  =       4                       ; RECIEVER ERROR
000100                          AVAIL   =       100                     ; AVAILABLE 1 = AVAILABLE
000400                          XMTERR  =       400                     ; TRANSMIT ERROR
100000                          RWRDY   =       100000                  ; IF SET, UDA IS ABLE TO READ AND/OR WRITE TO DRIVE
                                ;
                                ;       SDI COMMANDS AND RESPONSES
                                ;
000204                          DISCON  =       204                     ; DISCONNECT DRIVE
000006                          ERECOV  =       6                       ; ERROR RECOVERY
000201                          CHGMOD  =       201                     ; CHANGE MODE
000213                          DRVONL  =       213                     ; DRIVE ONLINE
000014                          DRVRUN  =       14                      ; DRIVE RUN
000005                          DRVCLR  =       5                       ; DRIVE CLEAR OPCODE
000207                          GETCHR  =       207                     ; GET CHARACTERISTICS
000210                          GETSUB  =       210                     ; GET SUBUNIT CHARACTERISTICS
000011                          GETSTA  =       11                      ; GET STATUS
000216                          IRECLB  =       216                     ; RECALIBRATE
000012                          INSEEK  =       12                      ; INITIATE SEEK
000176                          COMPLT  =       176                     ; SUCCESSFUL COMPLETION
000175                          UNSSUC  =       175                     ; UNSUCCESSFUL COMPLETION
000170                          CHRRES  =       170                     ; GET CHARACTERISTICS RESPONSE
000167                          SBCRES  =       167                     ; GET SUBUNIT CHARACTERISTICS RESPONSE
000366                          STSRES  =       366                     ; GET STATUS RESPONSE
000350                          ECHOC   =       350                     ; DIAGNOSTIC ECHO COMMAND AND RESPONSE
                                ;
                                ;       ERROR CODES
                                ;
000000                          FTLSYS  =       0                       ; SYSTEM FATAL ERROR
040000                          FTLDEV  =       40000                   ; DEVICE FATAL
100000                          ERHARD  =       100000                  ; HARD ERROR
140000                          ERSOFT  =       140000                  ; SOFT ERROR
```

```
                        .SBTTL  TEST 4 SPECIFIC INFORMATION
                        ;
                        ;       TEST 4 SPECIFIC INFORMATION
                        ;
                        ;
                        ;       CONSTANTS
                        ;
000377                  SCTWRD  =       255.                    ; NUMBER OF WORDS IN SECTOR TO FILL
000105                  INTEDC  =       69.                     ; INITIAL EDC VALUE
000061                  TLEN.U  =       U.LGRP+1                ; UNIT PARAMETER LENGTH
037716                  FIRSTU  =       HIMEM-TLEN.U            ; LOCATION OF FIRST UNIT PARAMETER BLOCK
                        ;
                        ;       UNIT PARAMETER OFFSETS
                        ;
000000                  U.NEXT  =       0.                      ; POINTER TO NEXT UNIT (RING LINKED LIST)
000001                  U.SUBP  =       U.NEXT+1                ; 4 WORDS OF SUBUNIT PARAMETER POINTERS
000005                  U.TIMO  =       U.SUBP+4                ; AREA TO STORE VARIOUS TIMEOUT VALUES
000006                  U.RWTO  =       U.TIMO+1                ; READ/WRITE TIMEOUT AREA
000007                  U.SEEK  =       U.RWTO+1                ; NUMBER OF SEEKS ISSUED
000010                  U.NFUN  =       U.SEEK+1                ; NEXT FUNCTION ADDRESS (FOR DEFERRED CALLS)
000011                  U.PAT   =       U.NFUN+1                ; PATTERN NUMBER TO WRITE
000012                  U.CCNT  =       U.PAT+1                 ; CURRENT COUNT OF T/G LOOPS
000014                  U.PCTG  =       U.CCNT+2                ; POINTER TO CURRENT TRACK OR GROUP
000015                  U.CTRK  =       U.PCTG+1                ; TRACK COUNT FOR GROUP OPERATIONS
000016                  U.NSEC  =       U.CTRK+1                ; NUMBER OF SECTORS R/W THIS TRY
000017                  U.MSEC  =       U.NSEC+1                ; NUMBER OF SECTORS TO BE R/W
000020                  U.TSEC  =       U.MSEC+1                ; NUMBER OF SECTORS TO BE R/W THIS OP
000021                  U.CSEC  =       U.TSEC+1                ; COUNT OF SECTORS R/W SO FAR
000022                  U.MASK  =       U.CSEC+1                ; UNIT MASK FOR XFC CALLS (0001 - 1000)
000023                  U.WRIT  =       U.MASK+1                ; WRITE PROTECTION STATUS
000024                  U.ELEV  =       U.WRIT+1                ; CURRENT ERROR RECOVERY LEVEL
000025                  U.RTRY  =       U.ELEV+1                ; MAXIMUM NUMBER OF READ RETRIES
000026                  U.MLEV  =       U.RTRY+1                ; MAXIMUM NUMBER OF ERROR RECOVERY LEVELS
000027                  U.ECCT  =       U.MLEV+1                ; ECC THRESHOLD
000030                  U.SDIS  =       U.ECCT+1                ; SDI SHORT TIMEOUT
000031                  U.SDIL  =       U.SDIS+1                ; SDI LONG TIMEOUT
000032                  U.WPRT  =       U.SDIL+1                ; MASK TO WRITE PROTECT READ-ONLY DRIVES
000033                  U.PARM  =       U.WPRT+1                ; UNIT PARAMETER WORD
000034                  U.SUBU  =       U.PARM+1                ; SUBUNIT OFFSET (0 - 3)
000035                  U.MBN   =       U.SUBU+1                ; MASTER L/DBN
000037                  U.CBN   =       U.MBN+2                 ; CURRENT L/DBN FOR START OF CHAIN
000041                  U.RBN   =       U.CBN+2                 ; RBN TO BE READ/WRITTEN IF LBN REVECTORED
000043                  U.COPY  =       U.RBN+2                 ; NUMBER OF RCT COPIES ON EACH SUBUNIT
000044                  U.CCOP  =       U.COPY+1                ; CURRENT RCT COPY THAT WE'RE WORKING ON
000045                  U.RWER  =       U.CCOP+1                ; ERROR (IF ANY) ON THE LAST R/W
000046                  U.RVER  =       U.RWER+1                ; ERROR THAT OCCURRED BEFOR THE REVECTOR OPERATION
000047                  U.SNUM  =       U.RVER+1                ; 4 WORDS THAT HOLD THE SUBUNIT LOGICAL NUMBERS
000053                  U.CCYL  =       U.SNUM+4                ; CURRENT CYLINDER
000055                  U.CGRP  =       U.CCYL+2
000056                  U.LCYL  =       U.CGRP+1                ; LAST CYLINDER SEEKED TO
000060                  U.LGRP  =       U.LCYL+2                ; LAST GROUP SEEKED TO
                        ;
                        ;       SUBUNIT PARAMETER OFFSET
                        ;
000000                  S.PARM  =       0.                      ; SUBUNIT PARAMETER WORD
000001                  S.SDCL  =       S.PARM+1                ; STARTING DIAGNOSTIC CYLINDER
000003                  S.PAT   =       S.SDCL+2                ; PATTERN TO USE FOR WRITES
```

```
     58          000004                S.TRKL  =       S.PAT+1                 ; NUMBER OF SECTORS IN ONE TRACK
     59          000005                S.SCHR  =       S.TRKL+1                ; POINTER TO SUBUNIT CHARACTERISTICS
     60          000006                S.MEGR  =       S.SCHR+1                ; SECTORS READ (UP TO 245)
     61          000007                S.MEGW  =       S.MEGR+1                ; SECTORS WRITTEN (UP TO 245)
     62          000010                S.BADP  =       S.MEGW+1                ; POINTER TO BAD BLOCK AREA
     63          000011                S.BESS  =       S.BADP+1                ; START OF BEGIN/END SETS
     64                                ;
     65                                ;       IF TRACK/GROUP LIMITS ARE GIVEN, THE SUBUNIT PARAMETERS HAVE THE
     66                                ;       FOLLOWING FIELDS ADDED TO THEM
     67                                ;
     68          000011                S.MCNT  =       S.BESS                  ; MAXIMUM TRACK/GROUP COUNT
     69          000013                S.TGOF  =       S.MCNT+2                ; ORIGINAL TRACK/GROUP OFFSET
     70          000015                S.TGSS  =       S.TGOF+2                ; START OF TRACK/GROUP SETS
```

```
                        ;
                        ;       DUMMY SDI CONTROL BLOCK OFFSETS
                        ;
000001                  D.LIMT  =       1                       ; DUMMY SDI SEARCH LIMIT
000002                  D.SCHR  =       2                       ; DUMMY POINTER TO SUBUNIT CHAR-5
                        ;
                        ;
                        ;       UNIT PARAMETER BITS (U.PARM(R5))
                        ;
100000                  DROP    =       100000                  ; DROP BIT (SET IF UNIT OR SUBUNIT DROPPED)
040000                  INITW   =       40000                   ; INITIAL WRITE (SET IF INITIAL WRITE IN PROG)
020000                  RESEEK  =       20000                   ; IF 1, INDICATES THAT A SEEK IS NECESSARY
010000                  DIREC   =       10000                   ; DIRECTION (SET IF SEQUENTIAL ACCESSES DECREASING)
004000                  NEWSUB  =       4000                    ; SET IF SEQUENTIAL SEEKS MOVED TO NEW SUBUNIT
002000                  SEKINP  =       2000                    ; SEEK IN PROGRESS - SET IF TRUE
001000                  FTIME   =       1000                    ; FIRST TIME FLAG - SET FOR INIT CODE
000400                  REVEC   =       400                     ; REVECTOR BIT (SET IF BLOCK REVECTORED)
000200                  RBNBN   =       200                     ; SEE IF WORKING ON RBN
000100                  REDWRT  =       100                     ; REDWRT (READ OR WRITE IN PROGRESS SET IF WRITE)
000040                  REVINP  =       40                      ; REVECTORING OPERATION IN PROGRESS
000020                  DATERR  =       20                      ; DATA ERROR IF SET
000004                  RETRY   =       4                       ; IF CLEAR, START RETRIES AT ZERO
000001                  RCLB    =       1                       ; RECALIBRATION BIT (SET IF RECALIBRATE JUST DONE)
                        ;
                        ;       SUBUNIT PARAMETER BITS (S.PARM(R4))
                        ;
020000                  DCYLS   =       20000                   ; DIAGNOSTIC CYLINDER FLAG (SET IF DBNS)
010000                  ECCCHK  =       10000                   ; 1 IF ECC CORRECTION ALLOWED   MASTER BITS
004000                  RONLY   =       4000                    ; READ ONLY (SET IF READ ONLY)
002000                  WONLY   =       2000                    ; WRITE ONLY (SET IF WRITE ONLY)
001000                  RTRIES  =       1000                    ; 1 IF RETRIES ALLOWED
000200                  ONLYCL  =       200                     ; SET IF ONLY CYLINDERS SPECIFIED
                                                                ; USED DURING SETUP ONLY
000100                  SEQSEK  =       100                     ; SEQUENTIAL SEEK (START UP TESTING ONLY)
000040                  BEUSED  =       40                      ; BEGIN/END SETS (USED IF SET)
000020                  TRACKS  =       20                      ; TRACKS OR GROUPS (TRACK IF SET)
000010                  WCHECK  =       10                      ; WRITE CHECK BIT (IF SET, WRITE CHECK WILL BE DONE)
                        ;                               WCHECK IS ALSO USED FOR THE UNIT PARAMETERS
000004                  WCHKAL  =       4                       ; SET IF WRITE CHECK ALWAYS TO BE DONE
000002                  DATCMP  =       2                       ; DATA COMPARE (SET IF DATA COMPARE TO BE DONE)
                        ;                               DATCMP IS ALSO USED FOR THE UNIT PARAMETERS
000001                  DCMPAL  =       1                       ; SET IF DATA COMPARE ALWAYS TO BE DONE
```

MACRO DEFINITIONS

```
                        .SBTTL  MACRO DEFINITIONS
                        ;
                        ;       DIAGNOSTIC MACRO FOR TEST4 OVERLAYS
                        ;
                                .MACRO  DIAG$$
                                TST     $$DIAG+$DIAG$
                                BEQ     .+6
                                MOV     #60000,R0
                                MOV     R0,@$$DIAG+$DIAG$
                                BR      .+1
                        $DIAG$  =       $DIAG$ + 1
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;       MESSAGE CONTROL TABLE MACRO
                        ;
                                .MACRO MSG CMDBUF,CMDSZ,RPLBUF,RPLSZ,SUCCOM
                                .WORD CMDBUF                    ;ADDRESS OF COMMAND
                                .WORD CMDSZ                     ;SIZE OF COMMAND IN BYTES
                                .WORD RPLBUF                    ;ADDRESS OF REPLY
                                .WORD RPLSZ                     ;SIZE OF REPLY IN WORDS
                                .IF NB NUMBER
                                .WORD SUCCOM                    ; SUCCESSFUL COMPLETION CODE
                                .ENDC
                                .ENDM
```

MACRO DEFINITIONS

```
      1                                .MACRO  BCS     LAB..
      2                                                        BCC     .+2
      3                                                        BR      LAB..
      4                                .ENDM
```

MACRO DEFINITIONS

```
                        ;       PUSH REGISTER MACRO
                        ;
                                .MACRO PUSH R9
                                .IRP X,<R9>

                                                        MOV X,-(SP)

                                .ENDR
                                .ENDM
                        ;
                        ;       POP REGISTER MACRO
                        ;
                                .MACRO POP R9
                                .IRP X,<R9>

                                                        MOV (SP)+,X

                                .ENDR
                                .ENDM
```

```
                        ;ERROR MACROS
                        ;THESE MACROS ARE CALLED TO REPORT ERRORS TO THE HOST PROGRAM.
                        ;THE MACRO NAMES ARE : ERRSF, ERRDF, ERRHRD, ERRSFT. EACH RESULTS IN THE HOST
                        ;BEING REQUESTED TO REPORT THE ERROR.
                        ;ARGUMENTS:     1 (MS$) MESSAGE POINTER
                        ;               2 (P1$) PARAMETER #1
                        ;               3 (P2$) PARAMETER #2
                        ;               4 (P3$) PARAMETER #3
                        ;               5 (P4$) PARAMETER #4
                        ;               6 (P5$) PARAMETER #5
                        ;               7 (P6$) PARAMETER #6
                        ;               8 (P7$) PARAMETER #7
                        ;               9 (P8$) PARAMETER #8
                        ;
                        ;THE MESSAGE POINTER MUST POINT TO AN ADDRESS IN THE OVERLAY 'MS'' IMMEDIATELY
                        ;FOLLOWING THE MAIN CODE. ANY ADDRESS MODE MAY BE USED (E.G. #MS1, @R2).
                        ;THE ADDRESS MUST CONTAIN AN ASCII FORMAT STRING TO DETERMINE THE MESSAGE
                        ;TO PRINT.
                        ;THE PARAMETER ARGUMENTS ARE OPTIONAL. THEY SHOULD BE SUPPLIED ONLY WHEN
                        ;THERE IS DATA TO BE PASSED TO THE HOST THAT WILL BE USED IN PRINTING THE
                        ;MESSAGE. THESE PARAMETER ARGUMENTS ARE THE ADDRESS OF DATA TO BE PASSED
                        ;USING ANY ADDRESSING MODE DESIRED.
                        ;ALL REGISTERS ARE RETURNED UNCHANGED. IT SHOULD BE NOTED THAT ARGUMENTS
                        ;CONTAINING SOMETHING OTHER THAN A REGISTER NAME (E.G.-#100 OR MEMADR)
                        ;ASSEMBLE TO INSTRUCTIONS THAT SAVE AND RESTORE A REGISTER ON THE STACK.

                                .MACRO ERRSF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ FTLSYS,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRDF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ FTLDEV,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRHRD MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NLIST
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ ERHARD,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .LIST
                                .ENDM

                                .MACRO ERRSFT MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ ERSOFT,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;THE FOLLOWING MACRO ACTUALLY PROCESSES THE ERROR CALL TO THE HOST PROGRAM

                        .MACRO ERROR$ ET$,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,ERRN$
                          .RADIX 8
                          PRMS=ARGS$-1
                          .IF LT,<PRMS>
                            .ERROR;NOT ENOUGH ARGUMENTS IN ERROR CALL
                          .ENDC
                          REGS$=-1
                          .IIF GE,<PRMS-8.>,PARGS. P8$
                          .IIF GE,<PRMS-7.>,PARGS. P7$
                          .IIF GE,<PRMS-6.>,PARGS. P6$
                          .IIF GE,<PRMS-5.>,PARGS. P5$
                          .IIF GE,<PRMS-4.>,PARGS. P4$
                          .IIF GE,<PRMS-3.>,PARGS. P3$
                          .IIF GE,<PRMS-2.>,PARGS. P2$
                          .IIF GE,<PRMS-1.>,PARGS. P1$
                          .IF GE REGS$
                            RSTR$ \REGS$
                          .ENDC
                          .RADIX 10
                          .LIST
                                                                        CALL RERROR     ;ERROR # ERRN$'.
                          .NLIST
                          .RADIX 8
                          .LIST
                                                                        .WORD ET$+ERRN
                                                                        .WORD <PRMS*10000>+MS$
                          .NLIST
                          ERRN=ERRN+1
                        .ENDM

                        .MACRO PARGS.,ADDR$
                          .NTYPE PTYPE$,ADDR$
                          .IF EQ,<PTYPE$&70>
                            .IIF EQ,<PTYPE$&7>-REGS$,RSTR$ \REGS$
                            .LIST
                                                                        MOV ADDR$,-(SP)
                            .NLIST
                          .IFF
                            .IF EQ,<PTYPE$&7>-1          ;PICK A REGISTER TO USE
                              REGU$=2                    ;SELECT R2 IF R1 IS USED IN PARAMETER FETCH
                            .IFF
                              REGU$=1                    ;OTHERWISE USE R1
                            .ENDC
                            .IF NE,<REGU$-REGS$>         ;IF REGISTER NOT ALREADY SAVED
                              .IF GE,REGS$
                                RSTR$ \REGS$             ;RESTORE CURRENT SAVED REGISTER
                              .ENDC
                              SAVR$ \REGU$               ;THEN SAVE SELECTED REGISTER
                            .ENDC
                            GETP$ \REGS$,ADDR$
                          .ENDC
                        .ENDM
```

MACRO DEFINITIONS

```
      1                                 .MACRO SAVR$ REGN
      2                                   .LIST
      3                                                       MOV R'REGN,SAVREG
      4                                   .NLIST
      5                                   REGS$=REGN
      6                                 .ENDM
      7
      8                                 .MACRO RSTR$ REGN
      9                                   .LIST
     10                                                       MOV SAVREG,R'REGN
     11                                   .NLIST
     12                                   REGS$=-1
     13                                 .ENDM
     14
     15                                 .MACRO GETP$ REGN,ADDR$
     16                                   .LIST
     17                                                       MOV ADDR$,R'REGN
     18                                                       MOV R'REGN,-(SP)
     19                                   .NLIST
     20                                 .ENDM
```

```
                        ;
                        ;       PRIMARY ERROR REPORTING (TEST 4)
                        ;
                        .MACRO  SOFTER  NUM,ARGS
                        ERROR   #ERSOFT,NUM,<ARGS>
                                                        MOV     #ERRMES,R2
                                                        MOV     R2,OUT.RQ
                        .ENDM

                        .MACRO  REPSFT  SFTFLG,ECCFG
                        .IF     NB,SFTFLG
                                                        MOV     #1,OUT.02
                        .IFF
                                                        CLR     OUT.02
                        .ENDC
                        .IF     NB,ECCFG
                                                        MOV     #1,OUT.03
                        .IFF
                                                        CLR     OUT.03
                        .ENDC
                                                        PUSH    R0
                                                        MOV     #U.SNUM,R0
                                                        ADD     R5,R0
                                                        ADD     U.SUBU(R5),R0
                                                        MOV     (R0),OUT.01
                                                        MOV     #T4SOFT,R0
                                                        CALL    HOSTRQ
                                                        POP     R0
                        .ENDM

                        .MACRO  HARDER  NUM,ARGS
                        ERROR   #ERHARD,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ
                        .ENDM

                        .MACRO  DEVFTL  NUM,ARGS
                        ERROR   #FTLDEV,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ
                        .ENDM

                        .MACRO  SYSFTL  NUM,ARGS
                        ERROR   #FTLSYS,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ
                        .ENDM

                        .MACRO  ERROR   TYPE,NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       5
                        MOVMSG  #ER'NUM,4
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
```

```
                        .ENDC

                                                        MOV     #NUM,OUT.02
                                                        BIS     TYPE,OUT.02
                                                        MOV     #.,OUT.01

                        .RADIX
                        .ENDM

                        .MACRO  CERROR  STNUM,ARGS
                        .RADIX  10
                NUMPTR  =       STNUM
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .RADIX
                        .ENDM

                        .MACRO  ERRORC  ARGS
                        .RADIX  10
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .RADIX
                        .ENDM

                        .MACRO  MOVMSG  ARG,INDX

                        .IF     LT,INDX-10
                                                        MOV     ARG,OUT.0'INDX
                        .IFF
                                                        MOV     ARG,OUT.'INDX
                        .ENDC
                        .ENDM

                        .MACRO  ENDERR  POS
                        .IF     NB,POS
                        NDERR   POS
                        .IFF
                        NDERR   \NUMPTR
                        .ENDC
                        .ENDM

                        .MACRO  NDERR   POS
                        .IF     NE,POS
                                                        BIS     #POS,ERRPOS     ; SET THE POSITION
                        .IFF
                                                        CLR     ERRPOS          ; CLEAR THE POSITION
                        .ENDC
                        .ENDM

                ;
                ;       MESSAGE REPORTING MACRO
                ;
                        .MACRO  MSSG    NUM,ARGS
                        .RADIX  10
                NUMPTR  =       3
                MOVMSG  #MS'NUM,2
```

```
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC

                                                        PUSH    <R0,R1>
                                                        MOV     #U.SNUM,R1
                                                        ADD     R5,R1
                                                        ADD     U.SUBU(R5),R1
                                                        MOV     (R1),OUT.01
                                                        MOV     #MESSAG,R0
                                                        CALL    HOSTRQ
                                                        POP     <R1,R0>

                        .RADIX
                        .ENDM
                ;
                ;
                        .MACRO  MSSGE   NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       3
                        MOVMSG  #'NUM,2
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC

                                                        PUSH    <R0,R1>
                                                        MOV     #MESSAG,R0
                                                        CALL    HOSTRQ
                                                        POP     <R1,R0>

                        .RADIX
                        .ENDM
```

MACRO DEFINITIONS

```
                        ;RETURN DRIVE STATUS MACRO WITH ERROR REPORTING

                                .MACRO  DSTAT,LAB$,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   ME
                                .LIST
                                CALL    RDSTAT                  ; GET DRIVE STATUS
                                BIT     #10000,R1               ; SEE IF ANY ERRORS
                                BEQ     2$                      ; IF NO ERROR, BRANCH
                                BIT     #4000,R1                ; SEE IF XMIT ERROR
                                BEQ     1$                      ; IF SO, BRANCH
                                ERRHRD  E1                      ; REPORT INVALID STATUS ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        1$:     ERRHRD  E2                      ; REPORT XMIT ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        2$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;
                        ;       XOR THE CONTENTS OF TWO REGISTERS
                        ;
                        .MACRO  RXOR    REG1,REG2
                        MOV     REG2,-(SP)              ; SAVE REGISTER REG2
                        BIC     REG1,REG2               ; CLEAR COORESPONDING BITS IN REG2
                        BIC     (SP)+,REG1              ; CLEAR COORESPONDING BITS IN REG1
                        BIS     REG1,REG2               ; OR WHAT'S LEFT
                        .ENDM
```

```
                        ;       SDI INTERCHANGE WITH DRIVE WITH ERROR REPORTING
                        
                                .MACRO  TALKX   ERRLAB,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   ME
                                .LIST
                                CALL    TALKER                  ; INITIATE SDI INTERCHANGE
                                TST     R3                      ; SEE IF ERROR OCCURRED
                                BEQ     12$                     ; IF NOT, BRANCH
                                BPL     11$                     ; IF SO, BRANCH
                                ERRHRD  E1;SEND COMMAND ERROR
                                BR      ERRLAB
                        11$:    ERRHRD  E2;RECEIVE COMMAND ERROR
                                BR      ERRLAB
                        12$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

START OF TEST CODE

```
                                .SBTTL  START OF TEST CODE
                                ;THE FOLLOWING IS FOR DEBUG PURPOSES ONLY. CAN BE NOP OR BREAKPOINT.
                                
 001364  000000                       .WORD   0                       ; FOR DMODT                     ;UDA52
                                ;INITIALIZE STACK
                                
 001365  104206  002101                 MOV #STACK,SP                   ;SET UP STACK POINTER
 001367                                 BR      START                   ; BRANCH OVER SUPPORT CODE
 001367  000000  002102                 ^O0,START
                                .SBTTL  RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
 001371                         RDSTAT:
                                ;
                                ;RETURN DRIVE STATUS
                                ;STATUS RETURNED IN DM REGISTER 1
                                ;
 001371                                 PUSH    <R3,R0>                 ; SAVE R3 AND R0
 001371  100463                                                                 MOV R3,-(SP)
 001372  100467                                                                 MOV R0,-(SP)
 001373  104203  000003                 MOV     #3,R3                   ; ALLOW ONLY 3 ERRORS
 001375  060007                 STATLP: XFC     STATUS                  ; GET DRIVE'S STATUS
 001376  103201  014000                 BIC     #14000,R1               ; CLEAR ERROR PASSING BITS
 001400  102201  000400                 BIT     #XMTERR,R1              ; CHECK XMIT ERRORS
 001402                                 BEQ     STATOK                  ; IF NO ERRORS, BRANCH
 001402  010000  001413                 ^O10000,STATOK
 001404  117403                        DEC     R3                      ; DECREMENT TRANSMIT ERROR COUNT
 0014C5                                 BNE     STATLP                  ; IF ERROR COUNT INCOMPLETE, BRANCH
 001405  050000  001375                 ^O50000,STATLP
 001407  104201  010000                 MOV     #10000,R1               ; FLAG AS TRANSMIT ERROR
 001411                                 BR      STATEX                  ; BRANCH
 001411  000000  001424                 ^O0,STATEX
 001413  102201  000004         STATOK: BIT     #RCVERR,R1              ; RECIEVER ERRORS
 001415                                 BNE     STATEX                  ; IF VALID, BRANCH
 001415  050000  001424                 ^O50000,STATEX
 001417  117403                        DEC     R3                      ; DECREMENT ERROR COUNT
 001420                                 BNE     STATLP                  ; IF ERROR COUNT NON-ZERO, BRANCH
 001420  050000  001375                 ^O50000,STATLP
 001422  104201  014000                 MOV     #14000,R1               ; FLAG AS INVALID STATUS ERROR
 001424                         STATEX: POP     <R0,R3>                 ; RESTORE R0, R3
 001424  104267                                                                 MOV (SP)+,R0
 001425  104263                                                                 MOV (SP)+,R3
 001426                                 RETURN                          ; RETURN TO CALLING MODULE
 001426  000000  000000                 ^O0,0
```

```
                                .SBTTL  HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
001430                          HOSTRQ:
                                ;
                                ;SEND REQUEST BUFFER TO HOST AND WAIT FOR RESPONSE.
                                ;CLEAR ARGUMENT AREA OF OUT BUFFER IN PREPARATION
                                ;FOR NEXT HOSTRQ CALL.
                                ;
                                ;INPUTS:
                                ;       R0 - HOST REQUEST NUMBER
                                ;       OUT BUFFER LOADED WITH DATA

001430                          PUSH <R0,R1,R2>
001430  100467                                                                  MOV R0,-(SP)
001431  100461                                                                  MOV R1,-(SP)
001432  100462                                                                  MOV R2,-(SP)
001433  104070  001470          MOV R0,OUT.RQ                   ;STORE REQUEST NUMBER IN BUFFER
001435  104207  001470  SNDAGN: MOV #OUT.RQ,R0                 ;SEND BUFFER TO HOST
001437  104201  000043          MOV #BUFSIZ,R1
001441  060016                  XFC MRD
001442  115001                  TST R1                          ;CHECK FOR ERROR
001443                          BNE SNDAGN                      ;IF ERROR, TRY AGAIN
001443  050000  001435          ^050000,SNDAGN
001445  104207  001533          MOV #IN.RQ,R0                   ;WAIT FOR RESPONSE FROM HOST
001447  104201  000043          MOV #BUFSIZ,R1
001451  060017                  XFC MWR
001452  104207  001471          MOV #OUT.01,R0                  ;CLEAR ARGUMENT WORDS IN BUFFER
001454  104201  000034          MOV #OUT.29-OUT.01,R1
001456  114002                  CLR R2
001457  100272  CLRBUF: MOV R2,(R0)+
001460  117401                  DEC R1
001461                          BPL CLRBUF
001461  030000  001457          ^030000,CLRBUF
001463                          POP <R2,R1,R0>
001463  104262                                                                  MOV (SP)+,R2
001464  104261                                                                  MOV (SP)+,R1
001465  104267                                                                  MOV (SP)+,R0
001466                          RETURN
001466  000000  000000          ^00,0
```

```
                                ;STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS
                                ;
                                ;OUT BUFFER - DATA TO SEND TO HOST
                                ;
001470  000000                  OUT.RQ: .WORD 0                 ;HOST REQUEST CODE
001471  000000                  OUT.01: .WORD 0                 ;DATA ARGUMENT  1
001472  000000                  OUT.02: .WORD 0                 ;DATA ARGUMENT  2
001473  000000                  OUT.03: .WORD 0                 ;DATA ARGUMENT  3
001474  000000                  OUT.04: .WORD 0                 ;DATA ARGUMENT  4
001475  000000                  OUT.05: .WORD 0                 ;DATA ARGUMENT  5
001476  000000                  OUT.06: .WORD 0                 ;DATA ARGUMENT  6
001477  000000                  OUT.07: .WORD 0                 ;DATA ARGUMENT  7
001500  000000                  OUT.08: .WORD 0                 ;DATA ARGUMENT  8
001501  000000                  OUT.09: .WORD 0                 ;DATA ARGUMENT  9
001502  000000                  OUT.10: .WORD 0                 ;DATA ARGUMENT 10
001503  000000                  OUT.11: .WORD 0                 ;DATA ARGUMENT 11
001504  000000                  OUT.12: .WORD 0                 ;DATA ARGUMENT 12
001505  000000                  OUT.13: .WORD 0                 ;DATA ARGUMENT 13
001506  000000                  OUT.14: .WORD 0                 ;DATA ARGUMENT 14
001507  000000                  OUT.15: .WORD 0                 ;DATA ARGUMENT 15
001510  000000                  OUT.16: .WORD 0                 ;DATA ARGUMENT 16
001511  000000                  OUT.17: .WORD 0                 ;DATA ARGUMENT 17
001512  000000                  OUT.18: .WORD 0                 ;DATA ARGUMENT 18
001513  000000                  OUT.19: .WORD 0                 ;DATA ARGUMENT 19
001514  000000                  OUT.20: .WORD 0                 ;DATA ARGUMENT 20
001515  000000                  OUT.21: .WORD 0                 ;DATA ARGUMENT 21
001516  000000                  OUT.22: .WORD 0                 ;DATA ARGUMENT 22
001517  000000                  OUT.23: .WORD 0                 ;DATA ARGUMENT 23
001520  000000                  OUT.24: .WORD 0                 ;DATA ARGUMENT 24
001521  000000                  OUT.25: .WORD 0                 ;DATA ARGUMENT 25
001522  000000                  OUT.26: .WORD 0                 ;DATA ARGUMENT 26
001523  000000                  OUT.27: .WORD 0                 ;DATA ARGUMENT 27
001524  000000                  OUT.28: .WORD 0                 ;DATA ARGUMENT 28
001525  000000                  OUT.29: .WORD 0                 ;DATA ARGUMENT 29
001526  000000                  OUT.30: .WORD 0                 ;DATA ARGUMENT 30
001527  000000                  OUT.31: .WORD 0                 ;DATA ARGUMENT 31
001530  000000                  OUT.32: .WORD 0                 ;DATA ARGUMENT 32
001531  000000                  OUT.33: .WORD 0                 ;DATA ARGUMENT 33
001532  000000                  OUT.34: .WORD 0                 ;DATA ARGUMENT 34
                                ;
                                ;IN BUFFER - DATA RECEIVED FROM HOST
                                ;
001533  000000                  IN.RQ:  .WORD 0                 ;HOST REQUEST CODE (ECHO)
001534  000000                  IN.01:  .WORD 0                 ;DATA ARGUMENT  1
001535  000000                  IN.02:  .WORD 0                 ;DATA ARGUMENT  2
001536  000000                  IN.03:  .WORD 0                 ;DATA ARGUMENT  3
001537  000000                  IN.04:  .WORD 0                 ;DATA ARGUMENT  4
001540  000000                  IN.05:  .WORD 0                 ;DATA ARGUMENT  5
001541  000000                  IN.06:  .WORD 0                 ;DATA ARGUMENT  6
001542  000000                  IN.07:  .WORD 0                 ;DATA ARGUMENT  7
001543  000000                  IN.08:  .WORD 0                 ;DATA ARGUMENT  8
001544  000000                  IN.09:  .WORD 0                 ;DATA ARGUMENT  9
001545  000000                  IN.10:  .WORD 0                 ;DATA ARGUMENT 10
001546  000000                  IN.11:  .WORD 0                 ;DATA ARGUMENT 11
001547  000000                  IN.12:  .WORD 0                 ;DATA ARGUMENT 12
001550  000000                  IN.13:  .WORD 0                 ;DATA ARGUMENT 13
001551  000000                  IN.14:  .WORD 0                 ;DATA ARGUMENT 14
```

```
     58 001552  000000                IN.15:  .WORD 0                 ;DATA ARGUMENT 15
     59 001553  000000                IN.16:  .WORD 0                 ;DATA ARGUMENT 16
     60 001554  000000                IN.17:  .WORD 0                 ;DATA ARGUMENT 17
     61 001555  000000                IN.18:  .WORD 0                 ;DATA ARGUMENT 18
     62 001556  000000                IN.19:  .WORD 0                 ;DATA ARGUMENT 19
     63 001557  000000                IN.20:  .WORD 0                 ;DATA ARGUMENT 20
     64 001560  000000                IN.21:  .WORD 0                 ;DATA ARGUMENT 21
     65 001561  000000                IN.22:  .WORD 0                 ;DATA ARGUMENT 22
     66 001562  000000                IN.23:  .WORD 0                 ;DATA ARGUMENT 23
     67 001563  000000                IN.24:  .WORD 0                 ;DATA ARGUMENT 24
     68 001564  000000                IN.25:  .WORD 0                 ;DATA ARGUMENT 25
     69 001565  000000                IN.26:  .WORD 0                 ;DATA ARGUMENT 26
     70 001566  000000                IN.27:  .WORD 0                 ;DATA ARGUMENT 27
     71 001567  000000                IN.28:  .WORD 0                 ;DATA ARGUMENT 28
     72 001570  000000                IN.29:  .WORD 0                 ;DATA ARGUMENT 29
     73 001571  000000                IN.30:  .WORD 0                 ;DATA ARGUMENT 30
     74 001572  000000                IN.31:  .WORD 0                 ;DATA ARGUMENT 31
     75 001573  000000                IN.32:  .WORD 0                 ;DATA ARGUMENT 32
     76 001574  000000                IN.33:  .WORD 0                 ;DATA ARGUMENT 33
     77 001575  000000                IN.34:  .WORD 0                 ;DATA ARGUMENT 34
     78         000043                BUFSIZ  =       . - IN.RQ       ;SIZE OF BUFFER
```

```
                                ;
                                ;       TALKER SENDS AND RECEIVES AN SDI INTERCHANGE
                                ;
                                ;INPUTS:
                                ;       R2 - SDI INTERCONNECT
                                ;       R3 - POINTER TO COMMAND TABLE CONTAINING APPROPRIATE COMMAND
                                ;       SDILTO/SDISTO SDI LONG AND SHORT TIMEOUTS, RESPECTIVELY
                                ;OUTPUTS:
                                ;       R0 - RETURN OP CODE FROM UNIT
                                ;       R3 - ERROR CODE - 0 = NO ERROR, 1 = RECEIVE ERROR, 100001 = SEND ERROR
                                ;
001576                          TALKER: PUSH    <R1,R4>         ; SAVE REGISTERS
001576  100461                                                                  MOV R1,-(SP)
001577  100464                                                                  MOV R4,-(SP)
001600  104237                          MOV     (R3)+,R0        ; SET ADR OF SDI COMMAND BUFFER
001601  104231                          MOV     (R3)+,R1        ; SET BUFFER LENGTH
001602  060004                          XFC     SEND            ; SEND COMMAND
001603  115001                          TST     R1              ; DID UNIT ACCEPT COMMAND
001604                                  BEQ     TALK1A          ; IF SO, BRANCH
001604  010000  001612                  ^O10000,TALK1A
001606  104203  100001                  MOV     #100001,R3      ; FLAG AS SEND ERROR
001610                                  BR      TALK2B          ; BRANCH TO EXIT
001610  000000  001652                  ^O0,TALK2B
001612  106203  002102          TALK1A: CMP     #LONG,R3        ; SEE IF LONG TIMEOUT TO BE USED
001614                                  BMI     1$              ; IF SO, BRANCH
001614  070000  001622                  ^O70000,1$
001616  104304  001656                  MOV     SDISTO,R4       ; SET UP SHORT TIMEOUT
001620                                  BR      TALK1B          ; BRANCH
001620  000000  001624                  ^O0,TALK1B
001622  104304  001657          1$:     MOV     SDILTO,R4       ; SET UP LONG TIMEOUT
001624  104137                  TALK1B: MOV     (R3),R0         ; SET DATA BUFFER ADDRESS
001625  104631  000001                  MOV     1(R3),R1        ; SET BUFFER LENGTH
001627  060005                          XFC     RCV             ; SEND RECEIVE SDI COMMAND
001630  115001                          TST     R1              ; DID ERROR OCCUR
001631                                  BEQ     TALK2A          ; IF NOT, BRANCH
001631  010000  001651                  ^O10000,TALK2A
001633  106201  000001                  CMP     #1,R1           ; SEE IF TIMEOUT
001635                                  BEQ     1$              ; IF SO, BRANCH
001635  010000  001642                  ^O10000,1$
001637  104013                          MOV     R1,R3           ; MOVE ERROR TYPE TO R3 FOR REPORTING
001640                                  BR      TALK2B          ; EXIT
001640  000000  001652                  ^O0,TALK2B
001642  117404                  1$:     DEC     R4              ; DECREMENT TIMEOUT VALUE
001643                                  BNE     TALK1B          ; IF NOT TIMEOUT, BRANCH
001643  050000  001624                  ^O50000,TALK1B
001645  104203  000001                  MOV     #1,R3           ; FLAG AS RECIEVE ERROR
001647                                  BR      TALK2B          ; BRANCH TO EXIT
001647  000000  001652                  ^O0,TALK2B
001651  114003                  TALK2A: CLR     R3              ; FLAG AS NO ERRORS
001652                          TALK2B: POP     <R4,R1>         ; RESTORE R4, R1
001652  104264                                                                  MOV (SP)+,R4
001653  104261                                                                  MOV (SP)+,R1
001654                                  RETURN
001654  000000  000000                  ^O0,0

001656  000012                  SDISTO: .WORD   10.             ; SDI SHORT TIMEOUT
001657  000024                  SDILTO: .WORD   20.             ; SDI LONG TIMEOUT
```

```
                                ;MREAD
                                ;
                                ;READ ONE WORD FROM UNIBUS MEMORY
                                ;INPUTS:
                                ;       R5 - ADDRESS OF WORD TO READ (LOW 16 BITS)
                                ;       R4 -    "            (HIGH 2 BITS)
                                ;OUTPUTS:
                                ;       R0 - DATA READ
                                ;
001660                          MREAD:  PUSH <R2,R3>                    ;SAVE SOME REGISTERS
001660  100462                                                                  MOV R2,-(SP)
001661  100463                                                                  MOV R3,-(SP)
001662  104057                          MOV R5,R0                       ;PUT UNIBUS ADDRESS IN R0 AND R1
001663  104041                          MOV R4,R1
001664  104202  000001                 MOV #1,R2                       ;TRANSFER ONE WORD
001666  104203  001720                 MOV #UDATA,R3                   ;LOCATION TO PUT DATA
001670  060013                          XFC UREAD                       ;DO THE READ
001671  104307  001720                 MOV UDATA,R0                    ;GET DATA READ
001673                                  POP <R3,R2>                     ;RESTORE OTHER REGISTERS
001673  104263                                                                  MOV (SP)+,R3
001674  104262                                                                  MOV (SP)+,R2
001675                                  RETURN
001675  000000  000000                 ^O0,0
```

```
                                ;MWRITE
                                ;
                                ;WRITE ONE WORD TO UNIBUS MEMORY
                                ;INPUTS:
                                ;       R5 - ADDRESS OF WORD TO WRITE (LOW 16 BITS)
                                ;       R4 -    "            (HIGH 2 BITS)
                                ;       R0 - DATA TO WRITE
                                ;OUTPUTS:

001677                          MWRITE: PUSH <R0,R2,R3>                 ;SAVE SOME REGISTERS
001677  100467                                                                  MOV R0,-(SP)
001700  100462                                                                  MOV R2,-(SP)
001701  100463                                                                  MOV R3,-(SP)
001702  104070  001720                  MOV R0,UDATA                   ;PUT DATA TO BE WRITTEN INTO BUFFER
001704  104057                          MOV R5,R0                      ;PUT UNIBUS ADDRESS IN R0 AND R1
001705  104041                          MOV R4,R1
001706  104202  000001                  MOV #1,R2                      ;TRANSFER ONE WORD
001710  104203  001720                  MOV #UDATA,R3                  ;ADDRESS OF DATA WORD
001712  060014                          XFC UWRITE                     ;DO THE WRITE
001713                                  POP <R3,R2,R0>                 ;RESTORE THE REGISTERS
001713  104263                                                                  MOV (SP)+,R3
001714  104262                                                                  MOV (SP)+,R2
001715  104267                                                                  MOV (SP)+,R0
001716                                  RETURN
001716  000000  000000                  ^O0,0

001720  000000                  UDATA:  .WORD 0                        ;DATA BUFFER FOR TRANSFERS TO AND FROM
                                                                       ;UNIBUS MEMORY
```

```
                                ;XOR
                                ;
                                ;PERFORM XOR LOGIC FUNCTION ON TWO REGISTERS
                                ;INPUTS:
                                ;       R1, R2 - DATA TO BE XOR'ED
                                ;OUTPUTS:
                                ;       R1 - UNCHANGED
                                ;       R2 - XOR OF TWO INPUTS
                                ;
 001721                         XOR:    PUSH R3                         ;SAVE R3
 001721  100463                                                                 MOV R3,-(SP)
 001722  104013                        MOV R1,R3
 001723  103023                        BIC R2,R3
 001724  103012                        BIC R1,R2
 001725  101032                        BIS R3,R2
 001726                         ;       POP R3                          ;RESTORE R3
 001726  104263                                                                 MOV (SP)+,R3
 001727  115002                        TST R2                          ;SET CONDITION CODES
 001730                         ;       RETURN
 001730  000000  000000                ^O0,0
```

```
                                TO:
                                ;
                                ;       CALCULATE THE TIMEOUT IN 9SEC INTERVALS (SDI RECEIVE XFC TAKES
                                ;       9 SEC)
                                ;
 1 001732                       ;
 2 001732  104201  000001       1$:     MOV     #1,R1           ; SET UP LOG2 SHIFTER
 3 001734  110201               2$:     ROL     R1              ; DOUBLE THE TIMEOUT VALUE
 4 001735  103201  000001               BIC     #1,R1           ; CLEAR THE LOW BIT
 5 001737  117407                       DEC     R0              ; DECREMENT COUNT
 6 001740                               BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
   001740  050000  001734               ^050000,1$
 7 001742  114007                       CLR     R0              ; CLEAR 9SEC COUNT
 8 001743  115407               2$:     INC     R0              ; INCREMENT 9 SEC COUNT
 9 001744  107201  000011               SUB     #9.,R1          ; SUBTRACT 9 SEC FROM TIMEOUT
10 001746                               BPL     2$              ; IF MORE TIME TO GO, BRANCH
   001746  030000  001743               ^030000,2$
11 001750                               RETURN                  ; RETURN TO CALLING PROGRAM
   001750  000000  000000               ^00,0
```

```
                                ;RERROR
                                ;
                                ;REPORT ERROR TO HOST PROGRAM
                                ;THIS ROUTINE IS CALLED BY THE ERROR MACROS:
                                ;ERRSF, ERRDF, ERRHRD AND ERRSFT
001752                          RERROR:
001752                                  PUSH R0                         ;SAVE ONE REGISTER
001752  100467                                                                  MOV R0,-(SP)
001753  104067                          MOV SP,R0                       ;GET STACK POINTER
001754                                  PUSH <R1,R2,R3,R4>              ;SAVE MORE REGISTERS
001754  100461                                                                  MOV R1,-(SP)
001755  100462                                                                  MOV R2,-(SP)
001756  100463                                                                  MOV R3,-(SP)
001757  100464                                                                  MOV R4,-(SP)
001760  115407                          INC R0                          ;CHANGE SAVED STACK POINTER TO POINT TO
                                                                        ; ADDRESS OF LOCATION AFTER CALL
001761  104271                          MOV (R0)+,R1                    ;GET RETURN PC
001762  104202  001471                  MOV #OUT.01,R2                  ;GET ADDRESS OF WHERE TO PUT DATA
001764  117401                          DEC R1                          ;REDUCE TO PC OF ERROR CALL
001765  100221                          MOV R1,(R2)+                    ;PUT PC IN OUT BUFFER
001766  115401                          INC R1                          ;GET BACK TO RETURN PC
001767  104213                          MOV (R1)+,R3                    ;GET ERROR NUMBER AND TYPE
001770  100223                          MOV R3,(R2)+                    ;PUT IN BUFFER
001771  104303  002037                  MOV LUNIT,R3                    ;PUT UNIT NUMBER IN BUFFER
001773  100223                          MOV R3,(R2)+
001774  104113                          MOV (R1),R3                     ;GET MESSAGE POINTER
001775  103203  170000                  BIC #^C007777,R3                ;CLEAR OTHER BITS
001777  100223                          MOV R3,(R2)+                    ;PUT IN OUT BUFFER
002000  104214                          MOV (R1)+,R4                    ;GET COUNT OF PARAMETERS
                                                                        ;R1 IS NOW POINTING TO INSTRUCTION AFTER ERROR CALL
002001  110704                          SWAB R4
002002  110604                          ROR R4                          ;EXTRACT NUMBER OF PARAMETERS FROM ERROR MACRO
002003  110604                          ROR R4
002004  110604                          ROR R4
002005  110604                          ROR R4
002006  103204  177760                  BIC #177760,R4
002010  104040  002034                  MOV R4,SPADJU+1                 ;SAVE FOR LATER ADJUSTMENT OF STACK POINTER
002012                                  BEQ RERRCA                      ;BRANCH IF NO PARAMETERS
002012  010000  002021                  ^O10000,RERRCA
002014  104273                  RERRPA: MOV (R0)+,R3                    ;GET PARAMETER
002015  100223                          MOV R3,(R2)+                    ;STORE PARAMETER IN OUT BUFFER
002016  117404                          DEC R4                          ;COUNT THE PARAMETERS
002017                                  BNE RERRPA                      ;GET NEXT IF MORE
002017  050000  002014                  ^O50000,RERRPA
002021  100471                  RERRCA: MOV R1,-(R0)                    ;PUT RETURN ADDRESS ON STACK
002022  104207  060013                  MOV #ERRMES,R0                  ;SEND ERROR PACKET TO HOST PROGRAM
002024                                  CALL HOSTRQ
002024  020000  001430                  ^O20000,HOSTRQ
002026                                  POP <R4,R3,R2,R1,R0>            ;RESTORE REGISTERS
002026  104264                                                                  MOV (SP)+,R4
002027  104263                                                                  MOV (SP)+,R3
002030  104262                                                                  MOV (SP)+,R2
002031  104261                                                                  MOV (SP)+,R1
002032  104267                                                                  MOV (SP)+,R0
002033  105206  000000          SPADJU: ADD #0,SP                       ;ADJUST STACK OVER PARAMETERS
                                                                        ;VALUE CHANGED ABOVE
002035                                  RETURN
```

```
        002035  000000  000000                  ^O0,0
    45
    46 002037  177777                  LUNIT:  .WORD -1                ;LOGICAL UNIT NUMBER (-1 FOR NOT AVAILABLE)
    47
    48 002040  000000                  SAVREG: .WORD 0                 ;STORAGE FOR REGISTER AT CALL TIME
```

```
                                ;STACK AREA
                        
002041  123456                          .WORD 123456            ;END MARKER FOR STACK
002042                                  .BLKW 31.               ;STACK
002101  123456          STACK:          .WORD 123456            ;MARKER FOR STACK UNDERFLOW
```

```
                                .SBTTL  FREE MEMORY CHECK

                        001750          ERRN=1000.                      ;START ERROR NUMBERS FROM 1000.

                                ;ASK HOST WHERE FREE MEMORY IS AND TO FILL IT WITH AN ADDRESS PATTERN

002102                          LONG:                                   ;UNUSED LABEL THAT MUST BE DEFINED
002102  104207  060000          START:  MOV #T1MSIZ,R0                  ;GET REQUEST NUMBER
002104                                  CALL HOSTRQ                     ;ASK HOST
002104  020000  001430                  ^O20000,HOSTRQ
002106  104207  001534                  MOV #IN.01,R0                   ;TRANSFER DATA FROM HOST
002110  104201  003252                  MOV #FWADR,R1                   ; TO STORAGE
002112  104202  000010                  MOV #8.,R2
002114  104273                  FMEMFL: MOV (R0)+,R3
002115  100213                          MOV R3,(R1)+
002116  117402                          DEC R2
002117                                  BNE FMEMFL
002117  050000  002114                  ^O50000,FMEMFL

                                ;READ ALL OF SPECIFIED MEMORY AND CHECK FOR DATA SAME AS ADDRESS

002121  104305  003252                  MOV FWADR,R5                    ;GET STARTING ADDRESS
002123  104304  003253                  MOV FWADRH,R4
002125                          ACHK1:  CALL MREAD                      ;READ FROM UNIBUS MEMORY
002125  020000  001660                  ^O20000,MREAD
002127  115001                          TST     R1                      ;WAS IT OK?
002130                                  BEQ     ACHK2                   ;IF SO, CONTINUE
002130  010000  002157                  ^O10000,ACHK2
002132  117401                          DEC     R1                      ;IF R1 = 1, NON EXITENT MEMORY
002133                                  BNE     1$                      ; IF NOT, CONTINUE
002133  050000  002145                  ^O50000,1$
002135                                  ERRHRD  MS1000,R5,R4
002135  100464                                                          MOV R4,-(SP)
002136  100465                                                          MOV R5,-(SP)
002137  020000  001752                  ^O20000,RERROR
002141  101750                                                          .WORD ERHARD+ERRN
002142  020000                                                          .WORD <PRMS*10000>+MS1000
002143                                  BR      ACHK3                   ;CONTINUE
002143  000000  002172                  ^O00,ACHK3
002145                          1$:     ERRHRD  MS1001,R5,R4,R5,R0
002145  100467                                                          MOV R0,-(SP)
002146  100465                                                          MOV R5,-(SP)
002147  100464                                                          MOV R4,-(SP)
002150  100465                                                          MOV R5,-(SP)
002151  020000  001752                  ^O20000,RERROR
002153  101751                                                          .WORD ERHARD+ERRN
002154  040063                                                          .WORD <PRMS*10000>+MS1001
002155                                  BR      ACHK3                   ;CONTINUE
002155  000000  002172                  ^O00,ACHK3

                                ;COMPARE DATA READ WITH EXPECTED

002157  106075          ACHK2:  CMP R0,R5                       ;COMPARE DATA READ WITH ADDRESS
002160                          BEQ ACHK3                       ;BRANCH IF A MATCH
002160  010000  002172          ^O10000,ACHK3
002162                          ERRHRD MS1002,R5,R4,R5,R0 ;UNIBUS ADDRESSING ERROR - INCORRECT DATA READ
002162  100467                                                          MOV R0,-(SP)
```

```
        002163  100465                                                  MOV R5,-(SP)
        002164  100464                                                  MOV R4,-(SP)
        002165  100465                                                  MOV R5,-(SP)
        002166  020000  001752          ^O20000,RERROR
        002170  101752                                                  .WORD ERHARD+ERRN
        002171  040170                                                  .WORD <PRMS*10000>+MS1002
    37
    38                          ;INCREMENT TO NEXT LOCATION UNTIL ALL TESTED
    39
    40 002172  106305  003254   ACHK3:  CMP LWADR,R5                    ;CHECK IF AT LAST ADDRESS
    41 002174                           BNE ACHK4
        002174  050000  002202          ^O50000,ACHK4
    42 002176  106304  003255           CMP LWADRH,R4
    43 002200                           BEQ BCHK                        ;GO TO NEXT TEST IF SO
        002200  010000  002211          ^O10000,BCHK
    44 002202  105205  000002   ACHK4:  ADD #2,R5                       ;INCREMENT TEST ADDRESS
    45 002204                           BCC ACHK1
        002204  040000  002125          ^O40000,ACHK1
    46 002206  115404                   INC R4                          ;CARRY TO HIGH BITS
    47 002207                           BR ACHK1                        ;LOOP IF STILL IN SPECIFIED MEMORY
        002207  000000  002125          ^O0,ACHK1
```

```
FREE MEMORY CHECK

                                        ;CHECK EACH ADDRESS LINE BY LOOKING AT TWO LOCATIONS WITH ONLY THAT
                                        ;ADDRESS LINE DIFFERENT AND VERIFYING TWO LOCATIONS ARE ACTUALLY ACCESSED

 002211  104307  003253       BCHK:   MOV FWADRH,R0           ;LOAD TEST ADDRESS HIGH BITS
 002213  104070  003263               MOV R0,TADRH
 002215  104203  000002               MOV #2,R3               ;GET STARTING ADDRESS BIT TO TEST
 002217  104032               BCHK1:  MOV R3,R2               ;GET ADDRESS BIT TO TEST
 002220  104301  003252               MOV FWADR,R1            ;GET FIRST ADDRESS OF FREE MEMORY
 002222                               CALL XOR                ;CHANGE JUST THE ONE BIT IN ADDRESS
 002222  020000  001721               ^O20000,XOR
 002224  104020  003262               MOV R2,TADR             ;STORE TEST ADDRESS
 002226                               CALL BCHKM              ;TEST THE MEMORY ADDRESS
 002226  020000  003101               ^O20000,BCHKM
 002230  105033                      ADD R3,R3               ;GET NEW ADDRESS BIT TO TEST
 002231                               BNE BCHK1               ;LOOP IF STILL AN ADDRESS BIT TO TEST
 002231  050000  002217               ^O50000,BCHK1

 002233  104307  003252               MOV FWADR,R0            ;NOW MOVE TO HIGH TWO BITS
 002235  104070  003262               MOV R0,TADR             ;COPY LOW BITS TO TEST ADDRESS
 002237  104203  000001               MOV #1,R3               ;START BIT
 002241  104032               BCHK2:  MOV R3,R2               ;GET TEST BIT
 002242  104301  003253               MOV FWADRH,R1           ;GET FIRST ADDRESS
 002244                               CALL XOR                ;CHANGE ONLY THE ONE BIT
 002244  020000  001721               ^O20000 XOR
 002246  104020  003263               MOV R2,TADRH            ;STORE AS TEST ADDRESS
 002250                               CALL BCHKM              ;TEST THE MEMORY ADDRESS
 002250  020000  003101               ^O20000,BCHKM
 002252  105033                      ADD R3,R3               ;CHANGE TEST BIT
 002253  102203  000004               BIT #4,R3               ;CHECK IF PAST TWO ADDRESS BITS
 002255                               BEQ BCHK2               ;REPEAT FOR OTHER BIT
 002255  010000  002241               ^O10000,BCHK2
```

```
                                ;TRANSFER LARGE BUFFERS OF DATA TO AND FROM THE HOST MEMORY.
                                ;
                                ;THE HOST MEMORY WILL BE DIVIDED INTO SEVERAL BUFFERS. ALL BUFFERS WILL
                                ;BE WRITTEN WITH PATTERN 0 THEN READ AND THE DATA COMPARED TO PATTERN 0.
                                ;EACH BUFFER WILL THEN BE WRITTEN TO PATTERN 1 WITH A READ OF ALL BUFFERS
                                ;AFTER EACH WRITE. WHEN ALL BUFFERS HAVE BEEN WRITTEN WITH PATTERN 1, THEN
                                ;THE SEQUENCE REPEATS BY WRITING EACH BUFFER WITH PATTERN 2 AND THEN
                                ;WITH PATTERN 3.
                                ;
                                ;DATA PATTERNS: (EACH IS A REPETITION OF THREE WORDS)
                                ;0 - 111111     1 - 177400      2 - 155555      3 - 000377
                                ;    044444         007760          133333          170017
                                ;    022222         000377          066666          177400

                                ;BREAK THE HOST MEMORY INTO BUFFERS

002257  104202  037777          CCHK:   MOV #HIMEM,R2                   ;COMPUTE SIZE OF DATA
002261  107202  003314                  SUB #FREE,R2                    ;  BUFFER IN M MEMORY
002263  105022                          ADD R2,R2                       ;CHANGE TO BYTE COUNT
                                ; *** FIND OUT IF WRITABLE HOST SPACE IS LESS THAN AVAILABLE DM SPACE
002264  104304  003254                  MOV     LWADR,R4                ;R4 = LAST WRITABLE HOST BUFFER WORD LOCATION
002266  104303  003255                  MOV     LWADRH,R3               ;R3 = EXTENDED ADDRESS BITS OF SAME
002270  107304  003252                  SUB     FWADR,R4                ;R4 = WRITABLE BUFFER SIZE
002272                                  BCC     1$                      ;IF DIDN'T CROSS PAGE BOUNDARY, CONTINUE
002272  040000  002275                  ^040000,1$
002274  117403                          DEC     R3                      ;ELSE, DECREMENT EXTENDED ADDRESS BITS BY 1
002275  107303  003253          1$:     SUB     FWADRH,R3               ;R3 = EXTENDED ADDRESS OF WRITABLE BUFFER SIZE
002277  105203  000000                  ADD     #0,R3                   ;CLEAR CARRY
                                ; *** DIVIDE WRITABLE REGION SIZE BY HALF
002301  110603                          ROR     R3                      ;ROTATE EXTENDED ADDRESS BITS INTO CARRY
002302  110604                          ROR     R4                      ;ROTATE EXTENDED ADDRESS IN & GET AREA/2
002303  102204  000001                  BIT     #BIT00,R4               ;MAKE SURE THE BYTE COUNT IS EVEN
002305                                  BEQ     2$                      ;IF IT IS, CONTINUE
002305  010000  002310                  ^010000,2$
002307  117404                          DEC     R4                      ;ELSE, FORCE BYTE COUNT TO BE EVEN
002310  115003          2$:     TST     R3                      ;IS R3 = 0?
002311                                  BNE     3$                      ;IF NOT, WRITABLE AREA >>> AVAILABLE DM BUFFER SPACE
002311  050000  002317                  ^050000,3$
002313  106042                          CMP     R4,R2                   ;IS WRITABLE AREA < AVAILABLE DM BUFFER SPACE
002314                                  BCC     3$                      ;IF NOT, CONTINUE
002314  040000  002317                  ^040000,3$
002316  104042                          MOV     R4,R2                   ;ELSE, DM SPACE > HOST WRITABLE SPACE
                                                                        ; STORE IN R2 & CONTINUE
002317  104205  003314          3$:     MOV #FREE,R5                    ;GET ADDRESS OF FIRST TABLE ENTRY
002321  104304  003252                  MOV FWADR,R4                    ;GET ADDRESS OF FIRST
002323  104303  003253                  MOV FWADRH,R3                   ;  BUFFER IN HOST
002325  114001                          CLR R1                          ;INIT COUNT OF BUFFERS
002326  100254          4$:     MOV R4,(R5)+                    ;STORE HOST BUFFER ADDRESS
002327  100253                  MOV R3,(R5)+                    ;  IN TABE ENTRY
002330  107202  000004                  SUB #4,R2                       ;REDUCE BUFFER SIZE BY TABLE ENTRY
002332  105024                          ADD R2,R4                       ;INCREADE HOST ADDRESS TO
002333                                  BCC 5$                          ;  NEXT BUFFER
002333  040000  002336                  ^040000,5$
002335  115403                          INC R3
002336  106303  003255          5$:     CMP LWADRH,R3                   ;CHECK IF BUFFER LARGER
002340                                  BMI 7$                          ;  THEN HOST MEMORY REMAINING
002340  070000  002355                  ^070000,7$
```

```
     52 002342                                  BNE 6$
        002342  050000  002352                 ^O50000,6$
     53 002344  106304  003254                 CMP LWADR,R4
     54 002346                                  BCC 6$
        002346  040000  002352                 ^O40000,6$
     55 002350                                  BR 7$
        002350  000000  002355                 ^O00,7$
     56 002352  115401                  6$:     INC R1                  ;COUNT THE TABLE ENTRY
     57 002353                                  BR 4$                   ;GO BACK AND DO AGAIN
        002353  000000  002326                 ^O00,4$
     58
     59 002355  104050  003267          7$:     MOV R5,DATBUF           ;SAVE ADDRESS OF DATA BUFFER
     60 002357  105207  000000                  ADD #0,R0               ;CLEAR CARRY
     61 002361  110602                          ROR R2                  ;SAVE SIZE OF DATA BUFFER
     62 002362  104020  003270                  MOV R2,SIZBUF           ;  IN WORDS
     63 002364  104010  003264                  MOV R1,BUFCNT           ;SAVE COUNT OF BUFFERS
```

```
                                ;WRITE ALL BUFFERS TO PATTERN 0

002366  114007                          CLR R0                  ;LOAD PATTERN 0
002367  104070  003266                  MOV R0,CURPAT           ;  IN SAVE WORD
002371  104070  003265          8$:     MOV R0,CURBUF           ;SELECT BUFFER 0
002373                                  CALL WRITE              ;WRITE THE BUFFER
002373  020000  002445                  ^O20000,WRITE
002375  104307  003265                  MOV CURBUF,R0           ;INCREMENT TO NEXT BUFFER
002377  115407                          INC R0
002400  106307  003264                  CMP BUFCNT,R0           ;WRITE ANOTHER IF
002402                                  BNE 8$                  ;  NOT AT LAST
002402  050000  002371                  ^O50000,8$
002404                                  CALL READ               ;READ ALL HOST BUFFERS
002404  020000  002525                  ^O20000,READ

002406  104307  003266          9$:     MOV CURPAT,R0           ;INCREMENT PATTERN NUMBER
002410  115407                          INC R0
002411  106207  000004                  CMP #4,R0               ;EXIT SUBTEST IF NO
002413                                  BEQ DONECD              ;  NO PATTERNS REMAINING
002413  010000  002437                  ^O10000,DONECD
002415  104070  003266                  MOV R0,CURPAT

002417  114007                          CLR R0                  ;POINT TO BUFFER 0
002420  104070  003265          10$:    MOV R0,CURBUF
002422                                  CALL WRITE              ;WRITE A BUFFER
002422  020000  002445                  ^O20000,WRITE
002424                                  CALL READ               ;READ ALL HOST BUFFERS
002424  020000  002525                  ^O20000,READ
002426  104307  003265                  MOV CURBUF,R0           ;INCREMENT TO NEXT BUFFER
002430  115407                          INC R0
002431  106307  003264                  CMP BUFCNT,R0           ;CHECK IF AT LAST
002433                                  BNE 10$                 ;  WRITE NEXT BUFFER
002433  050000  002420                  ^O50000,10$
002435                                  BR 9$                   ;  WRITE NEXT PATTERN
002435  000000  002406                  ^O0,9$
```

```
  002437  104207  060016          DONECD: MOV #DONE,RO                    ;END OF PROGRAM, TELL PDP-11
  002441                                  CALL HOSTRQ
  002441  020000  001430                  ^020000,HOSTRQ
  002443                                  BR DONECD                       ;KEEP SENDING IF RETURNED
  002443  000000  002437                  ^00,DONECD
```

```
                                ;FILL A DATA BUFFER BEGINNING AT ADDRESS IN DATBUF AND WHOSE SIZE
                                ;IS IN SIZBUF WITH A DATA PATTERN SPECIFIED BY CONTENTS OF CURPAT.
                                ;WRITE THIS BUFFER TO HOST STARTING AT ADDRESS IN TWO WORDS POINTED
                                ;TO BY CURBUF. THEN PLACE THE PATTERN NUMBER IN THE SECOND WORD OF THE
                                ;HOST ADDRESS

                                ;FILL BUFFER WITH DATA PATTERN

 002445  104307  003266   WRITE:  MOV CURPAT,R0            ;GET PATTERN NUMBER
 002447  105077                   ADD R0,R0               ;  TIMES FOUR
 002450  105077                   ADD R0,R0
 002451  104072                   MOV R0,R2               ;SAVE FOR ADDING TO TABLE ENTRY
 002452  105207  003062           ADD #PAT0,R0            ;ADD START OF PATTERN TABLES
 002454  104273                   MOV (R0)+,R3            ;GET PATTERN WORDS
 002455  104274                   MOV (R0)+,R4
 002456  104275                   MOV (R0)+,R5
 002457  104307  003267           MOV DATBUF,R0           ;GET ADDRESS OF BUFFER
 002461  104301  003270           MOV SIZBUF,R1           ;GET SIZE OF BUFFER

 002463  100273             1$:   MOV R3,(R0)+            ;LOAD ONE WORD
 002464  117401                   DEC R1                  ;COUNT THE WORDS
 002465                           BEQ 2$
 002465  010000  002477           ^010000,2$
 002467  100274                   MOV R4,(R0)+            ;LOAD ONE WORD
 002470  117401                   DEC R1                  ;COUNT THE WORDS
 002471                           BEQ 2$
 002471  010000  002477           ^010000,2$
 002473  100275                   MOV R5,(R0)+            ;LOAD ONE WORD
 002474  117401                   DEC R1                  ;COUNT THE WORDS
 002475                           BNE 1$
 002475  050000  002463           ^050000,1$

                                ;WRITE THE BUFFER TO HOST MEMORY

 002477  104305  003265     2$:   MOV CURBUF,R5           ;GET POINTER TO HOST ADDRESS
 002501  105055                   ADD R5,R5               ;  COUNT TIMES TWO
 002502  105205  003314           ADD #FREE,R5            ;  PLUS START ADDRESS
 002504  104257                   MOV (R5)+,R0            ;GET LOW ADDRESS BITS
 002505  104151                   MOV (R5),R1             ;GET HIGH ADDRESS BITS
 002506  103201  177774           BIC #177774,R1          ;CLEAR CURRENT PATTERN
 002510  104070  003303           MOV     R0,LBUFWL       ;SAVE LAST BUFFER WRITTEN
 002512  104010  003304           MOV     R1,LBUFWH
 002514  101012                   BIS R1,R2               ;SET IN NEW PATTERN
 002515  100152                   MOV R2,(R5)             ;STORE IN TABLE ENTRY
 002516  104302  003270           MOV SIZBUF,R2           ;GET WORDS TO TRANSFER
 002520  104303  003267           MOV DATBUF,R3           ;GET DM ADDRESS
 002522  060014                   XFC UWRITE              ;WRITE TO THE HOST MEMORY
 002523                           RETURN
 002523  000000  000000           ^00,0
```

```
                                ;READ AND PERFORM A DATA COMPARE ON ALL THE HOST BUFFERS. TWO WORD
                                ;TABLE ENTRIES STARTING AT FREE CONTAIN THE HOST ADDRESS OF THE
                                ;BUFFERS AND THE PATTERN NUMBER LAST WRITTEN TO THE BUFFER. THE
                                ;NUMBER OF BUFFERS IS IN BUFCNT AND THE SIZE OF THE BUFFERS IS IN
                                ;SIZBUF. THE DATA CAN BE READ INTO THE DM AT ADDRESS IN DATBUF.

 002525  104205  003314       READ:   MOV #FREE,R5                    ;GET ADDRESS OF TABLE ENTRIES
 002527  104304  003264               MOV BUFCNT,R4                   ;GET COUNT OF BUFFERS

                                ;READ DATA FROM HOST BUFFER INTO DM MEMORY

 002531  104303  003267       1$:     MOV     DATBUF,R3               ;R3 = ADDRESS OF DATA BUFFER IN DM MEMORY
 002533  104302  003270               MOV     SIZBUF,R2               ;SIZE OF BUFFER
 002535  104257                       MOV     (R5)+,R0                ;GET BUFFER ADDRESS
 002536  104151                       MOV     (R5),R1
 002537  103201  177774               BIC     #177774,R1              ;PLEAR PATTERN NUMBER
 002541  104070  003262               MOV     R0,TADR                 ;SAVE ADDRESS IN CASE OF ERROR
 002543  104010  003263               MOV     R1,TADRH
 002545  060013                       XFC     UREAD                   ;READ THE WORD
 002546  115001                       TST     R1                      ;WAS IT READ PROPERLY?
 002547                               BEQ     3$                      ;IF SO, GO ON TO DO MORE
 002547  010000  002620               ^010000,3$
 002551  117401                       DEC     R1                      ;WAS IT AN NXM ERROR?
 002552                               BNE     2$                      ;IF NOT, REPORT PARITY ERROR
 002552  050000  002577               ^050000,2$
 002554                               ERRHRD  MS1003,TADR,TADRH,SIZBUF
 002554  104010  002040                                                       MOV R1,SAVREG
 002556  104301  003270                                                       MOV SIZBUF,R1
 002560  100461                                                               MOV R1,-(SP)
 002561  104301  003263                                                       MOV TADRH,R1
 002563  100461                                                               MOV R1,-(SP)
 002564  104301  003262                                                       MOV TADR,R1
 002566  100461                                                               MOV R1,-(SP)
 002567  104301  002040                                                       MOV SAVREG,R1
 002571  020000  001752               ^020000,RERROR
 002573  101753                                                               .WORD ERHARD+ERRN
 002574  030332                                                               .WORD <PRMS*10000>+MS1003
 002575                               BR      3$                      ;EXIT
 002575  000000  002620               ^00,3$
 002577                       2$:     ERRHRD  MS1004,TADR,TADRH,SIZBUF
 002577  104010  002040                                                       MOV R1,SAVREG
 002601  104301  003270                                                       MOV SIZBUF,R1
 002603  100461                                                               MOV R1,-(SP)
 002604  104301  003263                                                       MOV TADRH,R1
 002606  100461                                                               MOV R1,-(SP)
 002607  104301  003262                                                       MOV TADR,R1
 002611  100461                                                               MOV R1,-(SP)
 002612  104301  002040                                                       MOV SAVREG,R1
 002614  020000  001752               ^020000,RERROR
 002616  101754                                                               .WORD ERHARD+ERRN
 002617  030452                                                               .WORD <PRMS*10000>+MS1004

                                ; *** SET UP OUT BUFFER IN CASE OF ERROR

 002620                       3$:     PUSH    <R5,R4>
 002620  100465                                                                 MOV R5,-(SP)
 002621  100464                                                                 MOV R4,-(SP)
```

```
                                        MOV #FREE,R5                    ;GET ADDRESS OF TABLE ENTRIES
                                        MOV BUFCNT,R4                   ;GET COUNT OF BUFFERS
                                        CMP     #4,R4                   ;IS R4 > 4?
                                        BCC     4$                      ;IF NOT, SKIP NEXT INSTRUCTION
002630  040000  002634                  ^040000,4$
                                        MOV     #4,R4                   ;IF SO, FORCE R4 = 4

                                ; *** STORE IN OUTPUT BUFFER JUST IN CASE OF ERROR
                                4$:     MOV     #OUT.05,R0              ;R0 -> OUTPUT BUFFER
                                        MOV     SIZBUF,R1
                                        MOV     R1,(R0)+                ;GIVE BUFFER SIZE
                                        DEC     R4                      ;ADJUST R4
                                        MOV     BTABLE(R4),R1           ;GIVE START OF ALL BUFFERS
                                        MOV     R1,(R0)+
                                        INC     R4                      ;RESTORE R4
                                5$:     MOV     (R5)+,R1                ;STORE BUFFER LO ADDRESS
                                        MOV     R1,(R0)+
                                        MOV     (R5)+,R1                ;STORE BUFFER HI ADDRESS
                                        BIC     #177774,R1
                                        MOV     R1,(R0)+
                                        DEC     R4                      ;DECREMENT (# OF BUFFERS)
                                        BNE     5$                      ;IF NOT DONE, CONTINUE
002655  050000  002646                  ^050000,5$
                                        MOV     R0,OUTPTR               ;OUT POINTER HAS ADDRESS PAST BUFFERS LOCATIONS
                                        ADD     #6,OUTPTR               ;OUTPRT -> PAST DATA OF FIXED LENGTH
                                        MOV     R0,OUTPT2               ;SAVE PREVIOUS POINTER
                                        POP     <R4,R5>
002666  104264                                                                 MOV (SP)+,R4
002667  104265                                                                 MOV (SP)+,R5

                                ;COMPARE DATA READ WITH DATA PATTERN

                                        MOV     SIZBUF,TEMP1            ;TEMP1 IS DECREMENT COUNTER
                                        MOV     DATBUF,R3               ;R3 -> DM DATA BUFFER
                                        DEC     R5                      ;R5 -> BACK TO LO BUFFER ADDRESS
                                        MOV     (R5)+,TADR              ;GET BUFFER ADDRESS
                                        MOV     (R5),TADRH              ;
                                        BIC     #177774,TADRH           ;MASK HI ADDRESS ONLY
                                        MOV     (R5)+,R2                ;R2 = PATTERN
                                        BIC     #177763,R2              ;MASK PATTERN
                                        ADD     #PAT0,R2                ;R2 -> PATTERN
                                        CLR     ERRNUM                  ;CLEAR ERROR COUNT
                                6$:     MOV     #3,TEMP2                ;TEMP2 = SIZE OF ANY PATTERN
                                7$:     MOV     (R3)+,R0                ;DATA RECEIVED
                                        MOV     (R2)+,R1                ;DATA EXPECTED
                                        CMP     R0,R1                   ;COMPARE THE TWO
                                        BEQ     8$                      ;IF EQUAL, NO ERROR
002722  010000  002753                  ^010000,8$

                                ; *** ERROR HERE

                                        INC     ERRNUM                  ;INCREASE THE ERROR COUNT
                                        CMP     #3,ERRNUM               ;MORE THAN 3 ERROR REPORTS?
                                        BMI     8$                      ;IF SO, DO NOT ADD ANY MORE TO THE ERROR REPORT
002731  070000  002753                  ^070000,8$
                                        PUSH    <R0,R1>
002733  100467                                                                 MOV R0,-(SP)
002734  100461                                                                 MOV R1,-(SP)
```

```
     80 002735  104307  003301                    MOV     OUTPTR,R0           ;R0 -> WHERE IN THE OUTPUT BUFFER
     81                                                                         ; TO PUT ERROR INFO
     82 002737  104301  003262                    MOV     TADR,R1
     83 002741  100271                            MOV     R1,(R0)+            ;STORE ERROR ADDRESS
     84 002742  104301  003263                    MOV     TADRH,R1
     85 002744  100271                            MOV     R1,(R0)+            ;
     86 002745  104261                            MOV     (SP)+,R1            ;STORE DATA EXPECTED
     87 002746  100271                            MOV     R1,(R0)+
     88 002747  104261                            MOV     (SP)+,R1            ;STORE DATA RECEIVED
     89 002750  100271                            MOV     R1,(R0)+
     90 002751  104070  003301                    MOV     R0,OUTPTR           ;STORE NEW OUTPUT POINTER
     91
     92                             ; *** RESUME NORMAL PROCESSING
     93
     94 002753  105200  000002  003262  8$:      ADD     #2,TADR             ;INCREMENT LO ADDRESS
     95 002756                                    BNE     9$                  ;IF DID NOT CROSS EXTENDED BOUNDARY, SKIP
        002756  050000  002762                    ^050000,9$
     96 002760  115400  003263                    INC     TADRH               ;ELSE, INCREMENT EXTENDED ADDRESS
     97 002762  117400  003277          9$:      DEC     TEMP1               ;GONE OVER THE ENTIRE BUFFER?
     98 002764                                    BEQ     10$                 ;IF SO, EXIT LOOP
        002764  010000  002776                    ^010000,10$
     99 002766  117400  003300                    DEC     TEMP2               ;AT END OF PATTERN?
    100 002770                                    BNE     7$                  ;IF NOT, LOOP BACK
        002770  050000  002717                    ^050000,7$
    101 002772  107202  000003                    SUB     #3,R2               ;R2 -> START OF PATTERN
    102 002774                                    BR      6$                  ;LOOP BACK TO RESET TEMP2
        002774  000000  002714                    ^00,6$
    103
    104                             ; *** ALL OF THE BUFFER HAS BEEN TESTED
    105
    106 002776  115000  003276          10$:     TST     ERRNUM              ;ANY ERRORS OCCURED?
    107 003000                                    BEQ     13$                 ;IF NOT, CHECK IF DONE WITH ROUTINE
        003000  010000  003055                    ^010000,13$
    108 003002  104307  003302                    MOV     OUTPT2,R0           ;R0 -> AFTER BUFFER LOCATIONS STORED
    109 003004  104021                            MOV     R2,R1               ;R2 -> PATTERN
    110 003005  107201  003062                    SUB     #PAT0,R1            ;ISOLATE PATTERN #
    111 003007  103201  177763                    BIC     #177763,R1          ;MASK PATTERN NUMBER
    112 003011  105201  000000                    ADD     #0,R1               ;CLEAR CARRY
    113 003013  110601                            ROR     R1
    114 003014  110601                            ROR     R1                  ;PATTERN NUMBER IN LO 2 BITS
    115 003015  100271                            MOV     R1,(R0)+            ;STORE PATTERN NUMBER
    116 003016  104301  003266                    MOV     CURPAT,R1
    117 003020  100271                            MOV     R1,(R0)+            ;LAST PATTERN WRITTEN
    118 003021  104301  003303                    MOV     LBUFWL,R1
    119 003023  100271                            MOV     R1,(R0)+            ;LAST BUFFER WRITTEN
    120 003024  104301  003304                    MOV     LBUFWH,R1
    121 003026  100271                            MOV     R1,(R0)+
    122 003027  104301  003276                    MOV     ERRNUM,R1
    123 003031  100271                            MOV     R1,(R0)+            ;ERROR COUNT
    124 003032  106200  000003  003276            CMP     #3,ERRNUM           ;DO WE HAVE MORE THAN 3 ERRORS?
    125 003035                                    BCC     11$                 ;IF NOT, CONTINUE
        003035  040000  003043                    ^040000,11$
    126 003037  104202  000002                    MOV     #2,R2               ;SET R2 FOR MAX
    127 003041                                    BR      12$                 ;
        003041  000000  003046                    ^00,12$
    128 003043  104302  003276          11$:     MOV     ERRNUM,R2           ;STORE ERROR COUNT IN R2
    129 003045  117402                            DEC     R2                  ;ADJUST R2
```

```
    130 003046  104621  003311          12$:    MOV     ETABLE(R2),R1           ;TABLE ENTRY
    131 003050  100171                          MOV     R1,(R0)
    132 003051                                  ERRHRD  MS1005
        003051  020000  001752                 ^O20000,RERROR
        003053  101755                                                          .WORD ERHARD+ERRN
        003054  000560                                                          .WORD <PRMS*10000>+MS1005
    133
    134                                 ;CYCLE THROUGH ALL HOST BUFFERS
    135
    136 003055  117404                  13$:    DEC     R4                      ;COUNT BUFFERS
    137 003056                                  BNE     1$                      ;REPEAT FOR ALL BUFFERS
        003056  050000  002531                 ^O50000,1$
    138 003060                          14$:    RETURN
        003060  000000  000000                 ^O0,0
    139
```

```
      1                                 ;DATA PATTERN TABLES
      2
      3 003062  111111                  PAT0:   .WORD ^B1001001001001001
      4 003063  044444                          .WORD ^B0100100100100100
      5 003064  022222                          .WORD ^B0010010010010010
      6 003065  000000                          .WORD 0
      7
      8 003066  177400                  PAT1:   .WORD ^B1111111100000000
      9 003067  007760                          .WORD ^B0000111111110000
     10 003070  000377                          .WORD ^B0000000011111111
     11 003071  000000                          .WORD 0
     12
     13 003072  155555                  PAT2:   .WORD ^B1101101101101101
     14 003073  133333                          .WORD ^B1011011011011011
     15 003074  066666                          .WORD ^B0110110110110110
     16 003075  000000                          .WORD 0
     17
     18 003076  000377                  PAT3:   .WORD ^B0000000011111111
     19 003077  170017                          .WORD ^B1111000000001111
     20 003100  177400                          .WORD ^B1111111100000000
```

```
                                        ;CHECK IF TEST ADDRESS IS IN BOUNDS OF READABLE MEMORY

003101  106300  003263  003257  BCHKM:  CMP TADRH,FRADRH        ;COMPARE TEST ADDRESS WITH FIRST READABE ADDRESS
003104                                  BEQ 1$                  ;BRANCH IF EQUAL
003104  010000  003112                  ^O10000,1$
003106                                  BCC 2$                  ;BRANCH IF TEST ADDRESS HIGHER
003106  040000  003121                  ^O40000,2$
003110                                  BR BCHKMX               ;BRANCH IF TEST ADDRESS LOWER
003110  000000  003250                  ^O0,BCHKMX
003112  106300  003262  003256  1$:     CMP TADR,FRADR
003115                                  BCC 2$                  ;BRANCH IF HIGHER OR SAME
003115  040000  003121                  ^O40000,2$
003117                                  BR BCHKMX               ;BRANCH IF LOWER
003117  000000  003250                  ^O0,BCHKMX
003121  106300  003261  003263  2$:     CMP LRADRH,TADRH        ;COMPARE TEST ADRESS WITH LAST READABLE ADDRESS
003124                                  BEQ 3$                  ;BRANCH IF EQUAL
003124  010000  003132                  ^O10000,3$
003126                                  BCC 4$                  ;BRANCH IF READABLE ADDRESS HIGHER
003126  040000  003141                  ^O40000,4$
003130                                  BR BCHKMX               ;BRANCH IF READABLE ADDRESS LOWER
003130  000000  003250                  ^O0,BCHKMX
003132  106300  003260  003262  3$:     CMP LRADR,TADR
003135                                  BCC 4$                  ;BRANCH IF HIGHER OR SAME
003135  040000  003141                  ^O40000,4$
003137                                  BR BCHKMX               ;BRANCH IF LOWER
003137  000000  003250                  ^O0,BCHKMX

003141  104305  003252          4$:     MOV FWADR,R5            ;WRITE ONES INTO FIRST ADDRESS
003143  104304  003253                  MOV FWADRH,R4
003145  104207  177777                  MOV #177777,R0
003147                                  CALL MWRITE
003147  020000  001677                  ^O20000,MWRITE
003151  104305  003262                  MOV TADR,R5             ;READ FROM TEST ADDRESS
003153  104304  003263                  MOV TADRH,R4
003155                                  CALL MREAD
003155  020000  001660                  ^O20000,MREAD
003157  106201  000001                  CMP     #1,R1           ;DID WE GET A NXM?
003161                                  BEQ     BCHKMX          ;IF SO, EXIT
003161  010000  003250                  ^O10000,BCHKMX
003163  106207  177777                  CMP #177777,R0          ;CHECK DATA READ FOR ALL ONES
003165                                  BNE BCHKMX              ;GO TO NEXT BIT IF NOT ALL ONES
003165  050000  003250                  ^O50000,BCHKMX

003167  104305  003252                  MOV FWADR,R5            ;WRITE ALL ZEROS TO FIRST ADDRESS
003171  104304  003253                  MOV FWADRH,R4
003173  114007                          CLR R0
003174                                  CALL MWRITE
003174  020000  001677                  ^O20000,MWRITE
003176  104305  003262                  MOV TADR,R5             ;READ FROM TEST ADDRESS
003200  104304  003263                  MOV TADRH,R4
003202                                  CALL MREAD
003202  020000  001660                  ^O20000,MREAD
003204  106201  000001                  CMP     #1,R1           ;DID WE GET A NXM?
003206                                  BEQ     BCHKMX          ;IF SO, EXIT
003206  010000  003250                  ^O10000,BCHKMX
003210  115007                          TST R0                  ;CHECK DATA READ FOR ALL ZEROS
003211                                  BNE BCHKMX              ;GO TO NEXT BIT IF NOT ALL ZEROS
```

FREE MEMORY CHECK

```
                003211  050000  003250                  ^050000,BCHKMX
     41
     42 003213  104301  003252                  MOV FWADR,R1                    ;COMPUTE XOR OF FIRST ADDRESS
     43 003215  104052                          MOV R5,R2                       ; AND TEST ADDRESS
     44 003216                                  CALL XOR
                003216  020000  001721          ^020000,XOR
     45 003220  104027                          MOV R2,R0                       ;RESULT TO R0
     46 003221  104301  003253                  MOV FWADRH,R1                   ;NOW DO IT FOR HIGH BITS
     47 003223  104042                          MOV R4,R2
     48 003224                                  CALL XOR
                003224  020000  001721          ^020000,XOR
     49 003226                                  ERRHRD MS1006,FWADR,FWADRH,R5,R4,R0,R2 ;UNIBUS ADDRESSING ERROR. TWO ADDRESSES READ
                003226  100462                                                          MOV R2,-(SP)
                003227  100467                                                          MOV R0,-(SP)
                003230  100464                                                          MOV R4,-(SP)
                003231  100465                                                          MOV R5,-(SP)
                003232  104010  002040                                                  MOV R1,SAVREG
                003234  104301  0032                                                   MOV FWADRH,R1
                003236  100461                                                          MOV R1,-(SP)
                003237  104301  003252                                                  MOV FWADR,R1
                003241  100461                                                          MOV R1,-(SP)
                003242  104301  002040                                                  MOV SAVREG,R1
                003244  020000  001752          ^020000,RERROR
                003246  101756                                                          .WORD ERHARD+ERRN
                003247  061105                                                          .WORD <PRMS*10000>+MS1006
     50
     51 003250                          BCHKMX: RETURN
                003250  000000  000000          ^00,0
```

```
                                ;PROGRAM VARIABLES

003252  000000                  FWADR:  .WORD 0                 ;FIRST ADDRESS CONTAINING ADDRESS DATA
003253  000000                  FWADRH: .WORD 0
003254  000000                  LWADR:  .WORD 0                 ;LAST ADDRESS CONTAINING ADDRESS DATA
003255  000000                  LWADRH: .WORD 0

003256  000000                  FRADR:  .WORD 0                 ;FIRST ADDRESS READABLE
003257  000000                  FRADRH: .WORD 0
003260  000000                  LRADR:  .WORD 0                 ;LAST ADDRESS READABLE
003261  000000                  LRADRH: .WORD 0

003262  000000                  TADR:   .WORD 0                 ;TEST ADDRESS
003263  000000                  TADRH:  .WORD 0

003264  000000                  BUFCNT: .WORD 0                 ;COUNT OF BUFFERS IN HOST MEMORY
003265  000000                  CURBUF: .WORD 0                 ;CURRENT BUFFER BEING WRITTEN IN HOST
003266  000000                  CURPAT: .WORD 0                 ;CURRENT DATA PATTERN BEING WRITTEN
003267  000000                  DATBUF: .WORD 0                 ;ADDRESS OF DATA BUFFER IN DM MEMORY
003270  000000                  SIZBUF: .WORD 0                 ;SIZE OF DATA BUFFER IN HOST MEMORY

                                ;HOST BUFFER TABLE ENTRIES ARE STORED AFTER THE PROGRAM AT FREE.
                                ;       TWO WORDS PER TABLE
                                ;               FIRST WORD  - LOW 16 BITS OF HOST ADDRESS
                                ;               SECOND WORD - BITS 1-0 HIGH TWO BITS OF HOST ADDRESS
                                ;                             BITS 3-2 PATTERN LAST WRITTEN

003271  000000                  CMPSIZ: .WORD 0                 ;TABLE FOR COMPARE DATA PATTERN XFC
003272  000000                  CMPADR: .WORD 0
003273  000000                  CMP1:   .WORD 0
003274  000000                  CMP2:   .WORD 0
003275  000000                  CMP3:   .WORD 0

                                ;       MISSELENEOUS DATA STORAGE
003276  000000                  ERRNUM: .WORD 0                 ;HOLDS NUMBER OF ERRORS DURING COMPARE
003277  000000                  TEMP1:  .WORD 0                 ;TEMPORARY STORAGE
003300  000000                  TEMP2:  .WORD 0                 ; SAME
003301  000000                  OUTPTR: .WORD 0                 ;POINTER INTO OUTPUT BUFFER FOR COMPARE
003302  000000                  OUTPT2: .WORD 0                 ; SAME
003303  000000                  LBUFWL: .WORD 0                 ;LAST BUFFER WRITTEN SAVED IN WRITE
003304  000000                  LBUFWH: .WORD 0                 ; SAME
003305  001274  001264  001254  BTABLE: .WORD B1,B2,B3,B4
003310  001244
003311  001357  001332  001305  ETABLE: .WORD E1,E2,E3
```

```
                                ;MESSAGE STORAGE OVERLAY

003314                          FREE:                                   ; FREE SPACE BEGINS HERE
003314                                  DMOVLY MS,0
003314  000105                         .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                        .NLIST BEX

000000      042     116     117 MS1000: .ASCII\'NON-EXISTANT MEMORY ERROR TRYING TO READ FROM UNIBUS.'N\
000034      123     062     065         .ASCII\S25'OCTAL'S6'HEX'N\
000045      123     070     042         .ASCII\S8'ADDRESS'S9018R9R9S5H18N\
000062      000                         .BYTE 0
000063      042     120     101 MS1001: .ASCII\'PARITY ERROR ON READ FROM UNIBUS.'N\
000105      123     062     065         .ASCII\S25'OCTAL'S6'HEX'N\
000116      123     070     042         .ASCII\S8'ADDRESS'S9018R9R9S5H18N\
000133      123     070     042         .ASCII\S8'DATA READ'S7016R9S5H16N\
000150      123     070     042         .ASCII\S8'DATA EXPECTED'S3016R9S5H16N\
000167      000                         .BYTE 0
000170      042     125     116 MS1002: .ASCII\'UNIBUS ADDRESSING ERROR - INCORRECT DATA READ.'N\
000220      042     115     105         .ASCII\'MEMORY LOCATION SHOULD CONTAIN OWN ADDRESS.'N\
000247      123     062     065         .ASCII\S25'OCTAL'S6'HEX'N\
000260      123     070     042         .ASCII\S8'ADDRESS'S9018R9R9S5H18N\
000275      123     070     042         .ASCII\S8'DATA READ'S7016R9S5H16N\
000312      123     070     042         .ASCII\S8'DATA EXPECTED'S3016R9S5H16N\
000331      000                         .BYTE 0
000332      042     116     117 MS1003: .ASCII\'NON-EXISTANT MEMORY ERROR TRYING TO READ FROM UNIBUS WITHIN BUFFER.'N\
000375      123     062     070         .ASCII\S28'OCTAL'S6'HEX'N\
000406      042     040     123         .ASCII\" STARTING ADDESS OF BUFFER '018R9R9S5H18N\
000433      123     070     042         .ASCII\S8'BUFFER SIZE'S8016R9S5H16N\
000451      000                         .BYTE 0
000452      042     120     101 MS1004: .ASCII\'PARITY ERROR ON READ FROM UNIBUS WITHIN BUFFER.'N\
000503      123     062     070         .ASCII\S28'OCTAL'S6'HEX'N\
000514      042     040     123         .ASCII\" STARTING ADDESS OF BUFFER '018R9R9S5H18N\
000541      123     070     042         .ASCII\S8'BUFFER SIZE'S8016R9S5H16N\
000557      000                         .BYTE 0
000560      042     104     101 MS1005: .ASCII\'DATA COMPARE FAILED AFTER WRITE THEN READ FROM UNIBUS.'N\
000614      042     040     102         .ASCII\" BUFFER SIZE = '016''(O)'R9S3H16''(X)'R9S3D16''.(D)'N\
000646      042     040     123         .ASCII\" STARTING ADDRESSES OF BUFFERS'N\
000666      123     066     042         .ASCII\S6'OCTAL'S11'HEX'NR1\
000700      042     040     103         .ASCII\" CURRENT DATA PATTERN READ'S17D6N\
000721      042     040     114         .ASCII\" LAST PATTERN WRITTEN'S22D6N\
000740      042     040     123         .ASCII\" STARTING ADDRESS OF LAST BUFFER WRITTEN'S3018''(O)'R9R9S3H18''(X)'N\
001001      042     040     116         .ASCII\" NUMBER OF ERRORS FOUND'S20D16''.(D)'N\
001024      123     064     042         .ASCII\S4'LOCATION'S6'DATA EXPECTED'S6'DATA RECEIVED'N\
001054      123     062     042         .ASCII\S2'OCTAL   HEX'S6'OCTAL   HEX'S8'OCTAL   HEX'N\
001103      122     061                 .ASCII\R1\
001104      000                         .BYTE 0
001105      042     125     116 MS1006: .ASCII\'UNIBUS ADDRESSING ERROR. TWO ADDRESSES READ SAME LOCATION.'N\
001143      123     063     063         .ASCII\S33'OCTAL'S6'HEX'N\
001154      123     070     042         .ASCII\S8'KNOWN GOOD ADDRESS'S6018R9R9S5H18N\
001177      123     070     042         .ASCII\S8'ERROR ADDRESS'S11018R9R9S5H18N\
001217      123     070     042         .ASCII\S8'ADDRESS BIT IN ERROR'S4018R9R9S5H18N\
001243      000                         .BYTE 0

001244      123     065     117 B4:     .ASCII\S5018R9R9S10H18N\
001254      123     065     117 B3:     .ASCII\S5018R9R9S10H18N\
001264      123     065     117 B2:     .ASCII\S5018R9R9S10H18N\
001274      123     065     117 B1:     .ASCII\S5018R9R9S10H18N\
```

FREE MEMORY CHECK

```
     57 001304      000                              .BYTE 0
     58
     59 001305      123      061      117  E3:       .ASCII\S1018S2R9R9H18S4016R9S2H16S7016R9S2H16S11N\
     60 001332      123      061      117  E2:       .ASCII\S1018S2R9R9H18S4016R9S2H16S7016R9S2H16S11N\
     61 001357      123      061      117  E1:       .ASCII\S1018S2R9R9H18S4016R9S2H16S7016R9S2H16S11N\
     62 001404      000                              .BYTE 0
     63
     64 001404                                       DMEND
        001405  000105                               .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
     65         000001                               .END
```

SYMBOL TABLE

```
ACHK1   002125        DRVID = 000004        HDRPRE= 000005        LARGE = 000001        OUT.21  001515
ACHK2   002157        DRVONL= 000213        HD.BAD= 110000        LBHINB= 177417        OUT.22  001516
ACHK3   002172        DRVRUN= 000014        HD.DBN= 140000        LBLONB= 177760        OUT.23  001517
ACHK4   002202        DU.DFL= 020000        HD.LBN= 000000        LBNCYL= 000000        OUT.24  001520
ARGS$ = 000007        DU.FTL= 050000        HD.PRV= 050000        LBNHST= 000012        OUT.25  001521
ATTN  = 000002        DU.INF= 030000        HD.RBN= 060000        LBNTRK= 000011        OUT.26  001522
AVAIL = 000100        DU.QUE= 010000        HD.REV= 030000        LBUFWH  003304        OUT.27  001523
BCHK    002211        DU.SPC= 060000        HD.XBN= 120000        LBUFWL  003303        OUT.28  001524
BCHKM   003101        DU.TER= 040000        HEADER= 000002        LINKLN= 000007        OUT.29  001525
BCHKMX  003250        D.LIMT= 000001        HIBYTE= 177400        LOBYTE= 000377        OUT.30  001526
BCHK1   002217        D.SCHR= 000002        HICYL = 000001        LONG    002102        OUT.31  001527
BCHK2   002241        ECC   = 000015        HIDBN = 000003        .ONGTO= 000001        OUT.32  001530
BEUSED= 000040        ECCCHK= 010000        HILBN = 000002        LOW   = 000002        OUT.33  001531
BF.DAT= 000000        ECCFLG= 010000        HIMEM = 037777        LRADR   003260        OUT.34  001532
BF.ECC= 000401        ECCRSH= 000002        HIRBN = 000003        LRADRH  003261        OVERFL= 000002
BF.EDC= 000400        ECHO  = 000010        HIXBN = 000002        LUNIT   002037        OVE.MN= 001364
BIT00 = 000001        ECHOC = 000350        HOSTRQ  001430        LWADR   003254        OVE.MS= 000000
BIT01 = 000002        EOC   = 100000        HRDREV= 000003        LWADRH  003255        OVL.MN= 001731
BREAK = 000000        ERECOV= 000006        INITW = 040000        MEDTYP= 000006        OVL.MS= 001406
BTABLE  003305        ERHARD= 100000        INSEEK= 000012        MESSAG= 060015        OVL...= 003337
BUFCNT  003264        ERLEV = 000002        INTEDC= 000105        MICREV= 000003        OVS.MN= 001040
BUFFLG= 040000        ERRMC = 060014        IN.RQ   001533        MRD   = 000016        OVS.MS= 004722
BUFSIZ= 000043        ERRMES= 060013        IN.01   001534        MREAD   001660        OV... = 003757
B1      001274        ERRN  = 001757        IN.02   001535        MS1000  000000        PAT0    003062
B2      001264        ERRNUM  003276        IN.03   001536        MS1001  000063        PAT1    003066
B3      001254        ERSOFT= 140000        IN.04   001537        MS1002  000170        PAT2    003072
B4      001244        ETABLE  003311        IN.05   001540        MS1003  000332        PAT3    003076
CCHK    002257        EXIT  = 000021        IN.06   001541        MS1004  000452        PHYSA = 001000
CHECK = 000010        E1      001357        IN.07   001542        MS1005  000560        PRMS  = 000006
CHGMOD= 000201        E2      001332        IN.08   001543        MS1006  001105        PTYPE$= 000030
CHRRES= 000170        E3      001305        IN.09   001544        MWR   = 000017        RBNBN = 000200
CLRBUF  001457        FB.DAT= 000000        IN.10   001545        MWRITE  001677        RBNTRK= 000004
CMPADR  003272        FB.EDC= 000400        IN.11   001546        NEWSUB= 004000        RBUFLN= 000415
CMPSIZ  003271        FCTSIZ= 000010        IN.12   001547        ONLYCL= 000200        RCLB  = 000001
CMP1    003273        FIRSTU= 037716        IN.13   001550        OUTPTR  003301        RCONT = 000000
CMP2    003274        FMEMFL  002114        IN.14   001551        OUTPT2  003302        RCTCPS= 000001
CMP3    003275        FORMAT= 000001        IN.15   001552        OUT.RQ  001470        RCTCSZ= 000014
COMPAR= 000006        FRADR   003256        IN.16   001553        OUT.01  001471        RCV   = 000005
COMPLT= 000176        FRADRH  003257        IN.17   001554        OUT.02  001472        RCVERR= 000004
CURBUF  003265        FRAME = 000004        IN.18   001555        OUT.03  001473        RCVRDY= 000001
CURPAT  003266        FREE    003314        IN.19   001556        OUT.04  001474        RDSTAT  001371
CVT   = 000020        FSTOP = 100000        IN.20   001557        OUT.05  001475        READ    002525
DATBUF  003267        FTIME = 001000        IN.21   001560        OUT.06  001476        REDWRT= 000100
DATCMP= 000002        FTLDEV= 040000        IN.22   001561        OUT.07  001477        REGS$ = 177777
DATERR= 000020        FTLSYS= 000000        IN.23   001562        OUT.08  001500        REGU$ = 000001
DATPRE= 000005        FT.BUF= 000000        IN.24   001563        OUT.09  001501        RERRCA  002021
DBNCYL= 000022        FT.HI = 000001        IN.25   001564        OUT.10  001502        RERROR  001752
DCMPAL= 000001        FT.LOW= 000002        IN.26   001565        OUT.11  001503        RERRPA  002014
DCYLS = 020000        FWADR   003252        IN.27   001566        OUT.12  001504        RESEEK= 020000
DINIT = 000011        FWADRH  003253        IN.28   001567        OUT.13  001505        RETRY = 000004
DIREC = 010000        GETCHR= 000207        IN.29   001570        OUT.14  001506        RETS  = 000001
DISCON= 000204        GETSTA= 000011        IN.30   001571        OUT.15  001507        REVEC = 000400
DONE  = 060016        GETSUB= 000210        IN.31   001572        OUT.16  001510        REVECT= 000004
DONECD  002437        GRPCYL= 000002        IN.32   001573        OUT.17  001511        REVINP= 000040
DROP  = 100000        GRPOFF= 000011        IN.33   001574        OUT.18  001512        REVS  = 000007
DRTYPE= 000007        HBHINB= 007777        IN.34   001575        OUT.19  001513        RM    = 000004
DRVCLR= 000005        HBLONB= 170377        IRECLB= 000216        OUT.20  001514        RONLY = 004000
```

SYMBOL TABLE

```
RREAL = 013400          ST.C  = 000002          S.TGOF= 000013          UDA52 = 000001          U.RBN = 000041
RSTOP = 100000          ST.CON= 000002          S.TGSS= 000015          UNIT0 = 000001          U.RTRY= 000025
RTRIES= 001000          ST.DB = 001000          S.TRKL= 000004          UNIT1 = 000002          U.RVER= 000046
RWRDY = 100000          ST.DF = 000020          TADR    003262          UNIT2 = 000004          U.RWER= 000045
RW.ANG= 000006          ST.DR = 000040          TADRH   003263          UNIT3 = 000010          U.RWTO= 000006
RW.BUF= 000001          ST.ERR= 000002          TALKER  001576          UNSSUC= 000175          U.SDIL= 000031
RW.CMD= 000004          ST.FE = 000200          TALK1A  001612          UREAD = 000013          U.SDIS= 000030
RW.HI = 000003          ST.FO = 002000          TALK1B  001624          UTOTST= 060012          U.SEEK= 000007
RW.LOW= 000002          ST.MOD= 000001          TALK2A  001651          UWRITE= 000014          U.SNUM= 000047
RW.SDI= 000005          ST.MSK= 000000          TALK2B  001652          U.CBN = 000037          U.SUBP= 000001
RW.STA= 000000          ST.OA = 000200          TEMP1   003277          U.CCNT= 000012          U.SUBU= 000034
SAVREG  002040          ST.PE = 000040          TEMP2   003300          U.CCOP= 000044          U.TIMO= 000005
SBCRES= 000167          ST.PS = 000002          TEST4 = 000000          U.CCYL= 000053          U.TSEC= 000020
SCTWRD= 000377          ST.RE = 000100          TIMEOU= 000001          U.CGRP= 000055          U.WPRT= 000032
SDILTO  001657          ST.RR = 000100          TLEN.U= 000061          U.COPY= 000043          U.WRIT= 000023
SDISTO  001656          ST.RTY= 000003          TO      001732          U.CSEC= 000021          WAITSI= 000012
SDIVER= 000000          ST.RU = 000001          TOOBIG= 000001          U.CTRK= 000015          WBUFLN= 000401
SEKINP= 002000          ST.SR = 000020          TRACKS= 000020          U.ECCT= 000027          WCHECK= 000010
SEND  = 000004          ST.STA= 000001          TRKGRP= 000003          U.ELEV= 000024          WCHKAL= 000004
SEQSEK= 000100          ST.S7 = 000400          T1MSIZ= 060000          U.LCYL= 000056          WCONT = 040000
SHRTTO= 000000          ST.UNT= 000000          T2CMD = 060002          U.LGRP= 000060          WONLY = 002000
SIZBUF  003270          ST.WE = 000010          T2DLL = 060001          U.MASK= 000022          WREAL = 122400
SNDAGN  001435          SUB   = 000013          T4BB1 = 060005          U.MBN = 000035          WRITE   002445
SPADJU  002033          S.BADP= 000010          T4BB2 = 060006          U.MLEV= 000026          WRONG = 000002
SS    = 000001          S.BESS= 000011          T4MPRM= 060003          U.MSEC= 000017          WSTOP = 140000
STACK   002101          S.MCNT= 000011          T4MXFR= 060011          U.NEXT= 000000          XBNCYL= 000021
START   002102          S.MEGR= 000006          T4SEEK= 060010          U.NFUN= 000010          XFERRT= 000000
STATEX  001424          S.MEGW= 000007          T4SOFT= 060007          U.NSEC= 000016          XMTERR= 000400
STATLP  001375          S.PARM= 000000          T4UPRM= 060004          U.PARM= 000033          XOR     001721
STATOK  001413          S.PAT = 000003          UDADM1= 001000 G        U.PAT = 000011          XREAD = 000002
STATUS= 000007          S.SCHR= 000005          UDATA   001720          U.PCTG= 000014          XWRITE= 000003
STSRES= 000366          S.SDCL= 000001          UDA50 = 000000

. ABS.  007736      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  16825 WORDS  ( 66 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  71 PAGES
,B:UDAT1.L52/C=[30,30]DMAC52/M,B:UDAT1
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
DONECD  32-16    33-1#    33-3
DROP     7-10#
DRTYPE   4-15#
DRVCLR   5-35#
DRVID    4-14#
DRVONL   5-33#
DRVRUN   5-34#
DU.DFL   2-50#
DU.FTL   2-53#
DU.INF   2-51#
DU.QUE   2-49#
DU.SPC   2-54#    5-3      5-4      5-5      5-6      5-7      5-8      5-9      5-10     5-11     5-12     5-13     5-14     5-15
         5-16     5-17
DU.TER   2-52#
E1      38-43    39-61#
E2      38-43    39-60#
E3      38-43    39-59#
ECC      3-16#
ECCCHK   7-28#
ECCFLG   2-88#
ECCRSH   4-11#
ECHO     3-11#
ECHOC    5-46#
EOC      2-89#
ERECOV   5-31#
ERHARD   5-52#    29-27    29-29    29-36    35-24    35-26    35-132   37-49
ERLEV    4-10#
ERRMC    5-15#
ERRMES   5-14#    27-39
ERRN    29-3#     29-27    29-27    29-27    29-27#   29-29    29-29    29-29    29-29#   29-36    29-36    29-36    29-36#   35-24
        35-24     35-24    35-24#   35-26    35-26    35-26    35-26#   35-132   35-132   35-132   35-132#  37-49    37-49    37-49
        37-49#
ERRNUM  35-67*    35-76*   35-77*   35-106*  35-122   35-124*  35-128   38-35#
ERSOFT   5-53#
ETABLE  35-130    38-43#
EXIT     3-20#
FB.DAT   2-65#
FB.EDC   2-66#
FCTSIZ   4-36#
FIRSTU   6-11#
FMEMFL  29-13#    29-16
FORMAT   3-4#
FRADR   37-7*     38-8#
FRADRH  37-3*     38-9#
FRAME    4-66#
FREE    31-18     31-40    34-34    35-7     35-31    39-3#
FSTOP    2-85#
FT.BUF   2-58#
FT.HI    2-59#
FT.LOW   2-60#
FTIME    7-16#
FTLDEV   5-51#
FTLSYS   5-50#
FWADR   29-11     29-20    30-8     30-15    31-23    31-41    37-18    37-30    37-42    37-49    37-49    38-3#
FWADRH  29-21     30-4     30-19    31-26    31-42    37-19    37-31    37-46    37-49    37-49    38-4#
GETCHR   5-36#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
GETSTA    5-38#
GETSUB    5-37#
GRPCYL    4-25#
GRPOFF    4-38#
HBHINB    4-56#
HBLONB    4-57#
HD.BAD    2-98#
HD.DBN    2-101#
HD.LBN    2-95#
HD.PRV    2-99#
HD.RBN    2-96#
HD.REV    2-97#
HD.XBN    2-100#
HDRPRE    4-34#
HEADER    4-62#
HIBYTE    4-54#
HICYL     4-24#
HIDBN     4-30#
HILBN     4-26#
HIMEM     2-36#     6-11     31-17
HIRBN     4-29#
HIXBN     4-27#
HOSTRQ   20-2#     27-40     2,-9      33-2
HRDREV    4-13#
IN.01    21-44#    29-10
IN.02    21-45#
IN.03    21-46#
IN.04    21-47#
IN.05    21-48#
IN.06    21-49#
IN.07    21-50#
IN.08    21-51#
IN.09    21-52#
IN.10    21-53#
IN.11    21-54#
IN.12    21-55#
IN.13    21-56#
IN.14    21-57#
IN.15    21-58#
IN.16    21-59#
IN.17    21-60#
IN.18    21-61#
IN.19    21-62#
IN.20    21-63#
IN.21    21-64#
IN.22    21-65#
IN.23    21-66#
IN.24    21-67#
IN.25    21-68#
IN.26    21-69#
IN.27    21-70#
IN.28    21-71#
IN.29    21-72#
IN.30    21-73#
IN.31    21-74#
IN.32    21-75#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
IN.33   21-76#
IN.34   21-77#
IN.RQ   20-19    21-43#   21-78
INITW    7-11#
INSEEK   5-40#
INTEDC   6-9#
IRECLB   5-39#
LARGE    4-74#
LBHINB   4-58#
LBLONB   4-59#
LBNCYL   4-23#
LBNHST   4-39#
LBNTRK   4-37#
LBUFWH  34-39*   35-120   38-41#
LBUFWL  34-38*   35-118   38-40#
LINKLN   2-113#
LOBYTE   4-55#
LONG    22-20    29-7#
LONGTO   4-6#
LOW      4-70#
LRADR   37-14    38-10#
LRADRH  37-10    38-11#
LUNIT   27-19    27-46#
LWADR   29-40    31-21    31-53    38-5#
LWADRH  29-42    31-22    31-50    38-6#
MEDTYP   4-35#
MESSAG   5-16#
MICREV   4-12#
MRD      3-17#   20-16
MREAD   23-10#   29-22    37-24    37-36
MS1000  29-27    39-7#
MS1001  29-29    39-11#
MS1002  29-36    39-17#
MS1003  35-24    39-24#
MS1004  35-26    39-29#
MS1005  35-132   39-34#
MS1006  37-49    39-46#
MWR      3-18#   20-21
MWRITE  24-10#   37-21    37-33
NEWSUB   7-14#
ONLYCL   7-32#
OUT.01  20-22    20-23    21-6#    27-13
OUT.02  21-7#
OUT.03  21-8#
OUT.04  21-9#
OUT.05  21-10#   35-37
OUT.06  21-11#
OUT.07  21-12#
OUT.08  21-13#
OUT.09  21-14#
OUT.10  21-15#
OUT.11  21-16#
OUT.12  21-17#
OUT.13  21-18#
OUT.14  21-19#
OUT.15  21-20#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
OUT.16  21-21#
OUT.17  21-22#
OUT.18  21-23#
OUT.19  21-24#
OUT.20  21-25#
OUT.21  21-26#
OUT.22  21-27#
OUT.23  21-28#
OUT.24  21-29#
OUT.25  21-30#
OUT.26  21-31#
OUT.27  21-32#
OUT.28  21-33#
OUT.29  20-23    21-34#
OUT.30  21-35#
OUT.31  21-36#
OUT.32  21-37#
OUT.33  21-38#
OUT.34  21-39#
OUT.RQ  20-13*   20-14    21-5#
OUTPT2  35-53*   35-108   38-39#
OUTPTR  35-51*   35-52*   35-80    35-90*   38-38#
OV...    2-11     2-11     2-11#   39-4     39-4     39-4     39-4#    39-64    39-64#
OVE.MN   2-11#   39-4
OVE.MS  39-4#    39-64
OVERFL   4-75#
OVL...   2-11     2-11#   39-4     39-4#    39-64    39-64#
OVL.MN   2-11     2-11    39-4     39-4     39-4#
OVL.MS  39-64    39-64    39-64#
OVS.MN   2-11#
OVS.MS  39-4#
PAT0    34-13    35-66    35-110   36-3#
PAT1    36-8#
PAT2    36-13#
PAT3    36-18#
PHYSA    2-11     2-11     2-11     2-11     2-11#
PRMS    29-27    29-27    29-27    29-27    29-27    29-27    29-27    29-27    29-27    29-27    29-27#   29-29    29-29    29-29
        29-29    29-29    29-29    29-29    29-29    29-29    29-29    29-29#   29-36    29-36    29-36    29-36    29-36    29-36
        29-36    29-36    29-36    29-36    29-36#   35-24    35-24    35-24    35-24    35-24    35-24    35-24    35-24    35-24
        35-24    35-24#   35-26    35-26    35-26    35-26    35-26    35-26    35-26    35-26    35-26    35-26    35-26#   35-132
        35-132   35-132   35-132   35-132   35-132   35-132   35-132   35-132   35-132   35-132#  37-49    37-49    37-49    37-49
        37-49    37-49    37-49    37-49    37-49    37-49    37-49#
PTYPE$  29-27    29-27    29-27    29-27    29-27#   29-27#   29-29    29-29    29-29    29-29    29-29    29-29    29-29    29-29
        29-29#   29-29#   29-29#   29-29#   29-36    29-36    29-36    29-36    29-36    29-36    29-36    29-36    29-36#   29-36#
        29-36#   29-36#   35-24    35-24    35-24    35-24    35-24    35-24    35-24#   35-24#   35-24#   35-26    35-26    35-26
        35-26    35-26    35-26    35-26#   35-26#   35-26#   37-49    37-49    37-49    37-49    37-49    37-49    37-49    37-49
        37-49    37-49    37-49    37-49    37-49#   37-49#   37-49#   37-49#   37-49#   37-49#
RBNBN    7-18#
RBNTRK   4-31#
RBUFLN   2-112#
RCLB     7-23#
RCONT    2-84#
RCTCPS   4-8#
RCTCSZ   4-40#
RCV      3-8#    22-27
RCVERR   5-23#   19-47
```

CROSS REFERENCE TABLE (CREF V04.00)

```
SNDAGN  20-14#   20-18
SPADJU  27-32*   27-42#
SS       4-9#
ST.C     3-30#
ST.CON   3-29#
ST.DB    3-47#
ST.DF    3-44#
ST.DR    3-37#
ST.ERR   3-28#
ST.FE    3-41#
ST.FO    3-46#
ST.MOD   3-27#
ST.MSK   3-25#
ST.OA    3-35#
ST.PE    3-43#
ST.PS    3-39#
ST.RE    3-42#
ST.RR    3-36#
ST.RTY   3-31#
ST.RU    3-40#
ST.S7    3-48#
ST.SR    3-38#
ST.STA   3-26#
ST.UNT   3-24#
ST.WE    3-45#
STACK   19-19    28-5#
START   19-30    29-8#
STATEX  19-46    19-48    19-52#
STATLP  19-39#   19-44    19-50
STATOK  19-42    19-47#
STATUS   3-10#   19-39
STSRES   5-45#
SUB      4-22#
T1MSIZ   5-3#    29-8
T2CMD    5-5#
T2DLL    5-4#
T4BB1    5-8#
T4BB2    5-9#
T4MPRM   5-6#
T4MXFR   5-12#
T4SEEK   5-11#
T4SOFT   5-10#
T4UPRM   5-7#
TADR    30-10*   30-16*   35-17*   35-24    35-24    35-26    35-26    35-61*   35-82    35-94*   37-7     37-14*   37-22    37-34
        38-13#
TADRH   30-5*    30-21*   35-18*   35-24    35-24    35-26    35-26    35-62*   35-63*   35-84    35-96*   37-3     37-10*   37-23
        37-35    38-14#
TALK1A  22-17    22-20#
TALK1B  22-23    22-25#   22-35
TALK2A  22-29    22-38#
TALK2B  22-19    22-33    22-37    22-39#
TALKER  22-12#
TEMP1   35-58*   35-97*   38-36#
TEMP2   35-68*   35-99*   38-37#
TEST4    2-1#
TIMEOU   4-61#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
TLEN.U    6-10#    6-11
TO       26-1#
TOOBIG    4-69#
TRACKS    7-36#
TRKGRP    4-28#
U.CBN     6-41#    6-42
U.CCNT    6-22#    6-23
U.CCOP    6-44#    6-45
U.CCYL    6-48#    6-49
U.CGRP    6-49#    6-50
U.COPY    6-43#    6-44
U.CSEC    6-28#    6-29
U.CTRK    6-24#    6-25
U.ECCT    6-34#    6-35
U.ELEV    6-31#    6-32
U.LCYL    6-50#    6-51
U.LGRP    6-10     6-51#
U.MASK    6-29#    6-30
U.MBN     6-40#    6-41
U.MLEV    6-33#    6-34
U.MSEC    6-26#    6-27
U.NEXT    6-15#    6-16
U.NFUN    6-20#    6-21
U.NSEC    6-25#    6-26
U.PARM    6-38#    6-39
U.PAT     6-21#    6-22
U.PCTG    6-23#    6-24
U.RBN     6-42#    6-43
U.RTRY    6-32#    6-33
U.RVER    6-46#    6-47
U.RWER    6-45#    6-46
U.RWTO    6-18#    6-19
U.SDIL    6-36#    6-37
U.SDIS    6-35#    6-36
U.SEEK    6-19#    6-20
U.SNUM    6-47#    6-48
U.SUBP    6-16#    6-17
U.SUBU    6-39#    6-40
U.TIMO    6-17#    6-18
U.TSEC    6-27#    6-28
U.WPRT    6-37#    6-38
U.WRIT    6-30#    6-31
UDA50     1-31#    1-46     2-14     2-39     19-10
UDA52     1-32#    1-37     2-3      2-34     19-5     19-21     38-4
UDADM1    2-11#
UDATA    23-14    23-16    24-11*   24-15    24-20#
UNIT0     4-46#
UNIT1     4-47#
UNIT2     4-48#
UNIT3     4-49#
UNSSUC    5-42#
UREAD     3-14#    23-15    35-19
UTOTST    5-13#
UWRITE    3-15#    24-16    34-44
WAITSI    3-13#
WBUFLN    2-111#   2-112
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
BCC       2-8#    29-45   31-24   31-37   31-48   31-54   35-34   35-125  37-5    37-8    37-12   37-15
BCS      10-1#
BEQ       2-8#    19-42   22-17   22-29   22-31   27-33   29-24   29-35   29-43   30-25   31-32   32-16   34-22   34-25
         35-21    35-72   35-98   35-107  37-4    37-11   37-26   37-38
BMI       2-8#    22-21   31-51   35-78
BNE       2-8#    19-44   19-48   19-50   20-18   22-35   26-10   27-37   29-16   29-26   29-41   30-13   31-35   31-52
         32-10    32-26   34-28   35-23   35-50   35-95   35-100  35-137  37-28   37-40
BPL       2-8#    20-27   26-14
BR        2-8#    19-30   19-46   22-19   22-23   22-33   22-37   29-28   29-30   29-47   31-55   31-57   32-27   33-3
         35-25    35-102  35-127  37-6    37-9    37-13   37-16
CALL      2-8#    27-40   29-9    29-22   29-27   29-29   29-36   30-9    30-11   30-20   30-22   32-6    32-11   32-21
         32-22    33-2    35-24   35-26   35-132  37-21   37-24   37-33   37-36   37-44   37-48   37-49
CERROR   15-65#
DEVFTL   15-37#
DIAG$$    8-5#
DMCODE    2-7#     2-11
DMEND     2-7#    39-64
DMODT     2-9#
DMOVLY    2-7#     2-11    39-4
DSTAT    16-3#
ENDERR   15-93#
ERRDF    12-33#
ERRHRD   12-39#   29-27   29-29   29-36   35-24   35-26   35-132  37-49
ERROR    15-49#
ERROR$   13-3#    29-27   29-29   29-36   35-24   35-26   35-132  37-49
ERRORC   15-75#
ERRSF    12-27#
ERRSFT   12-47#
GETP$    14-15#   35-24   35-24   35-24   35-26   35-26   35-26   37-49   37-49
HARDER   15-31#
JMP       2-8#    19-53   20-29   22-40   23-18   24-18   25-17   26-15   27-44   34-45   35-138  37-51
MOVMSG   15-84#
MSG       9-3#
MSSG     15-111#
MSSGE    15-133#
NDERR    15-101#
OVTERM    2-11     2-11#   2-11#   39-4    39-4#   39-64
PARGS.   13-33#   29-27   29-27   29-29   29-29   29-29   29-29   29-36   29-36   29-36   29-36   35-24   35-24   35-24
         35-26    35-26   35-26   37-49   37-49   37-49   37-49   37-49   37-49
POP      11-11#   19-52   20-28   22-39   23-17   24-17   25-15   27-41   35-54
PUSH     11-3#    19-37   20-12   22-12   23-10   24-10   25-10   27-7    27-9    35-30   35-79
REPSFT   15-10#
RETURN    2-8#    19-53   20-29   22-40   23-18   24-18   25-17   26-15   27-44   34-45   35-138  37-51
RSTR$    14-8#    35-24   35-26   37-49
RXOR     17-4#
SAVR$    14-1#    35-24   35-26   37-49
SOFTER   15-4#
```


TABLE OF CONTENTS

```
                                .TITLE  UDAT1   UNIBUS ADDRESSING
                        ;
                        ;
                        ; COPYRIGHT (C)  1981
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
000001                  UDA50=1
000000                  UDA52=0
                        ;THIS PROGRAM SHALL BE ASSEMBLED WITH THE PROGRAM DMACRO
                        ;USING A COMMAND LINE SIMILAR TO:
                        ;
                        ;  UDAT1.BIN,UDAT1/C=[1,2]DMACRO,UDAT1
                        ;
                        ;       VERSION 1.0 FOR RELEASE
                        ;
                        ;       VERSION 2.0 FOR NEXT UDA
                        ;
```

```
                        TEST4   =       0                       ; THIS IS NOT TEST4

000000                          DMCODE  UDADM1,0,714,3,0,1000

                        .SBTTL  UDA DM PROGRAM PARAMETERS

                                .LIST MEB
                        ;
                        ;       EQUATES
                        ;
000001                  BIT00 = 1
000002                  BIT01 = 2
                        ;
                        ;       HIGHEST USABLE LOCATION OF UDA MEMORY + 1
                        ;
010000                  HIMEM   =       10000                   ; HIGH MEMORY+1
007774                  OVSTRT  =       7774                    ; OVERLAY ADDRESS LOCATION
                        ;
                        ;       DUP PACKET VALUE
                        ;
010000                  DU.QUE = 10000                          ;QUESTION
020000                  DU.DFL = 20000                          ;DEFAULT QUESTION
030000                  DU.INF = 30000                          ;INFORMATION
040000                  DU.TER = 40000                          ;TERMINATOR
050000                  DU.FTL = 50000                          ;FATAL ERROR
060000                  DU.SPC = 60000                          ;SPECIAL
                        ;
                        ;       OFFSETS FOR FORMAT TRACK TABLE
                        ;
000000                  FT.BUF  =       0.                      ;BUFFER POINTER OFFSET
000001                  FT.HI   =       1.                      ;HI ORDER HEADER OFFSET
000002                  FT.LOW  =       2.                      ;LOW ORDER HEADER OFFSET
                        ;
                        ;
                        ;       OFFSETS FOR FORMAT TRACK BUFFER
                        ;
000000                  FB.DAT  =         0.                    ;FIRST DATA WORD OFFSET
000400                  FB.EDC  =       256.                    ;EDC WORD OFFSET
                        ;
                        ;
                        ;       OFFSETS FOR READ/WRITE I/O CHAIN TABLES
                        ;
000000                  RW.STAT =       0.                      ;STATUS AND NEXT BUFFER POINTER OFFSET
000001                  RW.BUF  =       1.                      ;POINTER TO DATA BUFFER
000002                  RW.LOW  =       2.                      ;HI ORDER EXPECTED HEADER
000003                  RW.HI   =       3.                      ;LOW ORDER EXPECTED HEADER
000004                  RW.CMD  =       4.                      ;SDI COMMAND AND HEAD ADDRESS
000005                  RW.SDI  =       5.                      ;DUMMY SDI CONTROL BLOCK POINTER
000006                  RW.ANG  =       6.                      ;THETA FROM INDEX
                        ;
                        ;       CONSTANTS FOR READ AND WRITE XFC'S
                        ;
140000                  WSTOP   =       140000                  ; LAST ENTRY IN CHAIN FOR WRITE
040000                  WCONT   =       40000                   ; WRITE CONTINUE
100000                  RSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR READ
000000                  RCONT   =       0                       ; READ CONTINUE
100000                  FSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR FORMAT
```

```
                                WREAL   =       122400          ; WRITE REAL TIME ECOMMAND
                                RREAL   =       13400           ; READ REAL TIME COMMAND
                                ECCFLG  =       10000           ; ECC ERROR IN BUFFER BIT
                                EOC     =       100000          ; END OF CHAIN
                                BUFFLG  =       40000           ; BUFFER FULL OR EMPTY FLAG
                                ;
                                ;
                                ;       HEADER CODES
                                ;
                                HD.LBN  =       000000          ;GOOD LBN
                                HD.RBN  =       060000          ;GOOD RBN, PERHAPS UNUSED
                                HD.REV  =       030000          ;REVECTORED LBN
                                HD.BAD  =       110000          ;BAD BLOCK
                                HD.PRV  =       050000          ;PRIMARY REVECTORED BLOCK
                                HD.XBN  =       120000          ;XBN BLOCK
                                HD.DBN  =       140000          ;DBN BLOCK

                                ;       OFFSETS FOR DATA BUFFERS

                                BF.DAT  =         0.            ;DATA
                                BF.EDC  =       256.            ;ERROR DETECTION CODE
                                BF.ECC  =       257.            ;LAST 17 ECC RESIDUES

                                ;       BUFFER AND READ/WRITE CHAIN LINK SIZES

                                WBUFLN  =       257.            ; WRITE BUFFER SIZE
                                RBUFLN  =       WBUFLN+12.      ; READ BUFFER SIZE
                                LINKLN  =       7.              ; LINK SIZE
```

```
                                ;       XFC DEFINITION EQUATES
                                ;
000000                          BREAK   =       0.              ;BREAKPOINT XFC CODE
000001                          FORMAT  =       1.              ;FORMAT TRACK XFC CODE
000002                          XREAD   =       2.              ;READ N SECTORS XFC CODE
000003                          XWRITE  =       3.              ;WRITE N SECTORS XFC CODE
000004                          SEND    =       4.              ;SEND SDI COMMAND XFC CODE
000005                          RCV     =       5.              ;RECEIVE SDI MESSAGE XFC CODE
000006                          COMPARE =       6.              ;COMPARE DATA PATTERN TO BUFFER
000007                          STATUS  =       7.              ;RETURN DRIVE STATUS XFC CODE
000010                          ECHO    =       8.              ;ECHO DATA TO DRIVE XFC CODE
000011                          DINIT   =       9.              ;DRIVE INITIALIZE XFC CODE
000012                          WAITSI  =       10.             ;WAIT FOR SECTOR OR INDEX PULSE
000013                          UREAD   =       11.             ;READ UNIBUS MEMORY XFC CODE
000014                          UWRITE  =       12.             ;WRITE UNIBUS MEMORY XFC CODE
000015                          ECC     =       13.             ;DO ECC ON BUFFER XFC CODE
000016                          MRD     =       14.             ;SEND BUFFER TO MAINTENANCE READ COMMAND
000017                          MWR     =       15.             ;GET BUFFER FROM MAINTENANCE WRITE COMMAND
000020                          CVT     =       16.             ;CONVERT TO PHYSICAL ADDRESS XFC CODE
000021                          EXIT    =       17.             ;TERMINATE DM PROGRAM XFC CODE
                                ;
                                ;       GET STATUS OFFSETS
                                ;
000000                          ST.UNT  =       0.              ;UNIT NUMBER
000000                          ST.MSK  =       0.              ;SUBUNIT MASK
000001                          ST.STA  =       1.              ;STATUS BYTE
000001                          ST.MOD  =       1.              ;MODE BYTE
000002                          ST.ERR  =       2.              ;ERROR BYTE
000002                          ST.CON  =       2.              ;CONTROLLER BYTE
000002                          ST.C    =       2.              ;C BITS
000003                          ST.RTY  =       3.              ;RETRY COUNT/FAILURE CODE
                                ;
                                ;       STATUS BIT DEFINITIONS
                                ;
000200                          ST.OA   =       200             ; ONLINE TO ANOTHER (SET IF DRIVE UNAVAILABLE)
000100                          ST.RR   =       100             ; READJUSTMENT BIT (SET IF RECALIBRATION REQUIRED)
000040                          ST.DR   =       40              ; DIAGNOSTIC REQUEST (SET IF DIAGNOSTIC REQUESTED)
000020                          ST.SR   =       20              ; SPINDLE READY (SET IF SPINDLE READY)
000002                          ST.PS   =       2               ; PORT SWITCH (SET IF PORT SWITCH IN)
000001                          ST.RU   =       1               ; RUN/STOP SWITCH (SET IF RUN/STOP SWITCH IN)
000200                          ST.FE   =       200             ; FATAL ERROR (SET IF FATAL ERROR OCCURRED)
000100                          ST.RE   =       100             ; RETRIABLE ERROR (SET IF RETRIABLE ERROR OCCURRED)
000040                          ST.PE   =       40              ; PROTOCOL ERROR (SET IF PROTOCOL ERROR OCCURRED)
000020                          ST.DF   =       20              ; INITIALIZATION FAILURE (SET IF INIT FAILED)
000010                          ST.WE   =       10              ; WRITE ENABLE (SET IF WRT ATTEMPTED ON PROT DISK)
002000                          ST.FO   =       2000            ; FORMATTING (SET IF FORMATTING ENABLED)
001000                          ST.DB   =       1000            ; DIAGNOSTIC CYLS (SET IF DIAG CYL ACCESS ENABLED)
000400                          ST.S7   =       400             ; SECTOR SIZE (SET FOR 576 BYTE SECTORS)
```

```
                        ;       GET COMMON CHARACTERISTICS OFFSETS
                        ;
000000  SHRTTO  =      0.       ;SHORT TIMEOUT <3:0>
000000  SDIVER  =      0.       ;SDI VERSION <7:4>
000000  XFERRT  =      0.       ;TRANSFER RATE <15:0>
000001  LONGTO  =      1.       ;LONG TIMEOUT <3:0>
000001  RETS    =      1.       ;RETRIES <7:4>
000001  RCTCPS  =      1.       ;F/RCT COPIES <11:8>
000001  SS      =      1.       ;SECTOR SIZE <15:15>
000002  ERLEV   =      2.       ;ERROR RETRY LEVELS <7:0>
000002  ECCRSH  =      2.       ;ECC THRESHOLD <15:8>
000003  MICREV  =      3.       ;MICROCODE REVISION NUMBER <7:0>
000003  HRDREV  =      3.       ;HARDWARE REVISION NUMBER <15:8>
000004  DRVID   =      4.       ;UNIQUE DRIVE ID <47:0>
000007  DRTYPE  =      7.       ;DRIVE TYPE IDENTIFIER <7:0>
000007  REVS    =      7.       ;REVS/SECOND <15:8>
                        ;
                        ;       GET SUBUNIT CHARACTERISTICS OFFSETS
                        ;
                        ;THESE OFFSETS ARE CURRENTLY GIVEN AS FOLLOWING THE COMMON CHARACTERISTICS
                        ;
000013  SUB     =      11.      ;OFFSET TO PUT SUBUNIT AFTER COMMON;
000000  LBNCYL  =      0.       ;NUMBER OF CYLINDERS IN LBN AREA <31:0>
000001  HICYL   =      1.       ;HI ORDER CYLINDER BITS <15:12>
000002  GRPCYL  =      2.       ;GROUPS PER CYLINDER <7:0>
000002  HILBN   =      2.       ;HI STARTING LBN <11:8>
000002  HIXBN   =      2.       ;HI STARTING XBN <15:12>
000003  TRKGRP  =      3.       ;TRACKS PER GROUP <7:0>
000003  HIRBN   =      3.       ;HI STARTING RBN <11:8>
000003  HIDBN   =      3.       ;HI STARTING DBN <15:12>
000004  RBNTRK  =      4.       ;RBNS PER TRACK <6:0>
000004  RM      =      4.       ;REMOVABLE MEDIA <7:7> 1=REMOVEABLE
000005  DATPRE  =      5.       ;DATA PREAMBLE SIZE IN WORDS <7:0>
000005  HDRPRE  =      5.       ;HEADER PREAMBLE SIZE IN WORDS <15:8>
000006  MEDTYP  =      6.       ;MEDIA TYPE <31:0>
000010  FCTSIZ  =      8.       ;FCT COPY SIZE <15:0>
000011  LBNTRK  =      9.       ;LBNS PER TRACK <7:0>
000011  GRPOFF  =      9.       ;GROUP OFFSET (SECTORS) <15:8>
000012  LBNHST  =     10.       ;LBNS IN HOST AREA <31:0>
000014  RCTCSZ  =     12.       ;RCT COPY SIZE <15:0>
000021  XBNCYL  =     17.       ;CYLS IN XBN AREA <15:0>
000022  DBNCYL  =     18.       ;CYLS IN DBN AREA <15:8>
                        ;
                        ;       UNIT CODES
                        ;
000001  UNIT0   =      1.       ;UNIT ZERO CODE
000002  UNIT1   =      2.       ;UNIT ONE CODE
000004  UNIT2   =      4.       ;UNIT TWO CODE
000010  UNIT3   =      8.       ;UNIT THREE CODE
                        ;
                        ;       BIT MASK DEFINITIONS
                        ;
                        ;
177400  HIBYTE  =    177400     ;HIGH BYTE MASK
000377  LOBYTE  =    000377     ;LOW BYTE MASK
007777  HBHINB  =      7777     ;HI BYTE, HI NIBBLE MASK
170377  HBLONB  =    170377     ;HI BYTE, LO NIBBLE MASK
```

```
      58          177417                LBHINB  =       177417          ;LO BYTE, HI NIBBLE MASK
      59          177760                LBLONB  =       177760          ;LO BYTE, LO NIBBLE MASK
      60                                ;
      61          000001                TIMEOUT =       1.              ;DRIVE TIMEOUT CODE
      62          000002                HEADER  =       2.              ;HEADER COMPARE FAILURE CODE
      63          000004                REVECT  =       4.              ;REVECTOR NEEDED CODE
      64                                ;
      65          000002                WRONG   =       2.              ;FIRST WORD NOT START FRAME CODE
      66          000004                FRAME   =       4.              ;FRAMING ERROR CODE
      67          000010                CHECK   =       8.              ;CHECKSUM ERROR CODE
      68                                ;
      69          000001                TOOBIG  =       1.              ;NUMBER OF WORDS EXCEEDS 7064
      70          000002                LOW     =       2.              ;DM BUFFER ADDRESS IS LESS THAN 714
      71                                ;
      72                                ;       BIT MASK
      73                                ;
      74          000001                LARGE   =       1.              ;BLOCK NUMBER TOO LARGE
      75          000002                OVERFL  =       2.              ;SECTOR NUMBER LARGER THAN 16 BITS
```

```
                        ;MAINTANENCE READ/WRITE REQUEST NUMBERS
                        ;
060000                  T1MSIZ  =       0.+DU.SPC       ;GET FREE MEMORY PARAMETERS
060001                  T2DLL   =       1.+DU.SPC       ;DOWNLINE LOAD DRIVE DIAGNOSTIC
060002                  T2CMD   =       2.+DU.SPC       ;MANUAL INTERVENTION TEST 2 PROTOCOL
060003                  T4MPRM  =       3.+DU.SPC       ;GET MASTER PARAMETERS FROM SW QUESTIONS
060004                  T4UPRM  =       4.+DU.SPC       ;GET UNIT PARAMETERS FROM HW QUESTIONS
060005                  T4BB1   =       5.+DU.SPC       ;GET BAD BLOCKS (1 THRU 14)
060006                  T4BB2   =       6.+DU.SPC       ;GET REST OF BAD BLOCKS (15 AND 16)
060007                  T4SOFT  =       7.+DU.SPC       ;ADD TO SOFT ERROR AND ECC COUNT
060010                  T4SEEK  =       8.+DU.SPC       ;ADD 1 TO SEEK COUNT
060011                  T4MXFR  =       9.+DU.SPC       ;ADD TO MEGABITS READ AND WRITTEN
060012                  UTOTST  =       10.+DU.SPC      ;GET UNITS TO TEST
060013                  ERRMES  =       11.+DU.SPC      ;PRINT ERROR MESSAGE
060014                  ERRMC   =       12.+DU.SPC      ;TEST 4 ERROR REPORTING
060015                  MESSAG  =       13.+DU.SPC      ;INFORMATION MESSAGE
060016                  DONE    =       14.+DU.SPC      ;MARK DM PROGRAM AS NO LONGER RUNNING
                        ;
                        ;       OTHER BIT DEFINITIONS
                        ;
000001                  RCVRDY  =       1               ; RECIEVER READY 1 = READY
000002                  ATTN    =       2               ; ATTENTION BIT FOR RETURN DRIVE SIGNALS XFC
000004                  RCVERR  =       4               ; RECIEVER ERROR
000100                  AVAIL   =       100             ; AVAILABLE 1 = AVAILABLE
000400                  XMTERR  =       400             ; TRANSMIT ERROR
100000                  RWRDY   =       100000          ; IF SET, UDA IS ABLE TO READ AND/OR WRITE TO DRIVE
                        ;
                        ;       SDI COMMANDS AND RESPONSES
                        ;
000204                  DISCON  =       204             ; DISCONNECT DRIVE
000006                  ERECOV  =       6               ; ERROR RECOVERY
000201                  CHGMOD  =       201             ; CHANGE MODE
000213                  DRVONL  =       213             ; DRIVE ONLINE
000014                  DRVRUN  =       14              ; DRIVE RUN
000005                  DRVCLR  =       5               ; DRIVE CLEAR OPCODE
000207                  GETCHR  =       207             ; GET CHARACTERISTICS
000210                  GETSUB  =       210             ; GET SUBUNIT CHARACTERISTICS
000011                  GETSTA  =       11              ; GET STATUS
000216                  IRECLB  =       216             ; RECALIBRATE
000012                  INSEEK  =       12              ; INITIATE SEEK
000176                  COMPLT  =       176             ; SUCCESSFUL COMPLETION
000175                  UNSSUC  =       175             ; UNSUCCESSFUL COMPLETION
000170                  CHRRES  =       170             ; GET CHARACTERISTICS RESPONSE
000167                  SBCRES  =       167             ; GET SUBUNIT CHARACTERISTICS RESPONSE
000366                  STSRES  =       366             ; GET STATUS RESPONSE
000350                  ECHOC   =       350             ; DIAGNOSTIC ECHO COMMAND AND RESPONSE
                        ;
                        ;       ERROR CODES
                        ;
000000                  FTLSYS  =       0               ; SYSTEM FATAL ERROR
040000                  FTLDEV  =       40000           ; DEVICE FATAL
100000                  ERHARD  =       100000          ; HARD ERROR
140000                  ERSOFT  =       140000          ; SOFT ERROR
```

```
                        .SBTTL  TEST 4 SPECIFIC INFORMATION
                        ;
                        ;       TEST 4 SPECIFIC INFORMATION
                        ;
                        ;
                        ;       CONSTANTS
                        ;
000377                  SCTWRD  =       255.                    ; NUMBER OF WORDS IN SECTOR TO FILL
000105                  INTEDC  =       69.                     ; INITIAL EDC VALUE
000061                  TLEN.U  =       U.LGRP+1                ; UNIT PARAMETER LENGTH
007717                  FIRSTU  =       HIMEM-TLEN.U            ; LOCATION OF FIRST UNIT PARAMETER BLOCK
                        ;
                        ;       UNIT PARAMETER OFFSETS
                        ;
000000                  U.NEXT  =       0.                      ; POINTER TO NEXT UNIT (RING LINKED LIST)
000001                  U.SUBP  =       U.NEXT+1                ; 4 WORDS OF SUBUNIT PARAMETER POINTERS
000005                  U.TIMO  =       U.SUBP+4                ; AREA TO STORE VARIOUS TIMEOUT VALUES
000006                  U.RWTO  =       U.TIMO+1                ; READ/WRITE TIMEOUT AREA
000007                  U.SEEK  =       U.RWTO+1                ; NUMBER OF SEEKS ISSUED
000010                  U.NFUN  =       U.SEEK+1                ; NEXT FUNCTION ADDRESS (FOR DEFERRED CALLS)
000011                  U.PAT   =       U.NFUN+1                ; PATTERN NUMBER TO WRITE
000012                  U.CCNT  =       U.PAT+1                 ; CURRENT COUNT OF T/G LOOPS
000014                  U.PCTG  =       U.CCNT+2                ; POINTER TO CURRENT TRACK OR GROUP
000015                  U.CTRK  =       U.PCTG+1                ; TRACK COUNT FOR GROUP OPERATIONS
000016                  U.NSEC  =       U.CTRK+1                ; NUMBER OF SECTORS R/W THIS TRY
000017                  U.MSEC  =       U.NSEC+1                ; NUMBER OF SECTORS TO BE R/W
000020                  U.TSEC  =       U.MSEC+1                ; NUMBER OF SECTORS TO BE R/W THIS OP
000021                  U.CSEC  =       U.TSEC+1                ; COUNT OF SECTORS R/W SO FAR
000022                  U.MASK  =       U.CSEC+1                ; UNIT MASK FOR XFC CALLS (0001 - 1000)
000023                  U.WRIT  =       U.MASK+1                ; WRITE PROTECTION STATUS
000024                  U.ELEV  =       U.WRIT+1                ; CURRENT ERROR RECOVERY LEVEL
000025                  U.RTRY  =       U.ELEV+1                ; MAXIMUM NUMBER OF READ RETRIES
000026                  U.MLEV  =       U.RTRY+1                ; MAXIMUM NUMBER OF ERROR RECOVERY LEVELS
000027                  U.ECCT  =       U.MLEV+1                ; ECC THRESHOLD
000030                  U.SDIS  =       U.ECCT+1                ; SDI SHORT TIMEOUT
000031                  U.SDIL  =       U.SDIS+1                ; SDI LONG TIMEOUT
000032                  U.WPRT  =       U.SDIL+1                ; MASK TO WRITE PROTECT READ-ONLY DRIVES
000033                  U.PARM  =       U.WPRT+1                ; UNIT PARAMETER WORD
000034                  U.SUBU  =       U.PARM+1                ; SUBUNIT OFFSET (0 - 3)
000035                  U.MBN   =       U.SUBU+1                ; MASTER L/DBN
000037                  U.CBN   =       U.MBN+2                 ; CURRENT L/DBN FOR START OF CHAIN
000041                  U.RBN   =       U.CBN+2                 ; RBN TO BE READ/WRITTEN IF LBN REVECTORED
000043                  U.COPY  =       U.RBN+2                 ; NUMBER OF RCT COPIES ON EACH SUBUNIT
000044                  U.CCOP  =       U.COPY+1                ; CURRENT RCT COPY THAT WE'RE WORKING ON
000045                  U.RWER  =       U.CCOP+1                ; ERROR (IF ANY) ON THE LAST R/W
000046                  U.RVER  =       U.RWER+1                ; ERROR THAT OCCURRED BEFOR THE REVECTOR OPERATION
000047                  U.SNUM  =       U.RVER+1                ; 4 WORDS THAT HOLD THE SUBUNIT LOGICAL NUMBERS
000053                  U.CCYL  =       U.SNUM+4                ; CURRENT CYLINDER
000055                  U.CGRP  =       U.CCYL+2
000056                  U.LCYL  =       U.CGRP+1                ; LAST CYLINDER SEEKED TO
000060                  U.LGRP  =       U.LCYL+2                ; LAST GROUP SEEKED TO
                        ;
                        ;       SUBUNIT PARAMETER OFFSET
                        ;
000000                  S.PARM  =       0.                      ; SUBUNIT PARAMETER WORD
000001                  S.SDCL  =       S.PARM+1                ; STARTING DIAGNOSTIC CYLINDER
000003                  S.PAT   =       S.SDCL+2                ; PATTERN TO USE FOR WRITES
```

```
     58          000004                  S.TRKL  =       S.PAT+1                 ; NUMBER OF SECTORS IN ONE TRACK
     59          000005                  S.SCHR  =       S.TRKL+1                ; POINTER TO SUBUNIT CHARACTERISTICS
     60          000006                  S.MEGR  =       S.SCHR+1                ; SECTORS READ (UP TO 245)
     61          000007                  S.MEGW  =       S.MEGR+1                ; SECTORS WRITTEN (UP TO 245)
     62          000010                  S.BADP  =       S.MEGW+1                ; POINTER TO BAD BLOCK AREA
     63          000011                  S.BESS  =       S.BADP+1                ; START OF BEGIN/END SETS
     64                                  ;
     65                                  ;       IF TRACK/GROUP LIMITS ARE GIVEN, THE SUBUNIT PARAMETERS HAVE THE
     66                                  ;       FOLLOWING FIELDS ADDED TO THEM
     67                                  ;
     68          000011                  S.MCNT  =       S.BESS                  ; MAXIMUM TRACK/GROUP COUNT
     69          000013                  S.TGOF  =       S.MCNT+2                ; ORIGINAL TRACK/GROUP OFFSET
     70          000015                  S.TGSS  =       S.TGOF+2                ; START OF TRACK/GROUP SETS
```

```
                        ;
                        ;       DUMMY SDI CONTROL BLOCK OFFSETS
                        ;
000001                  D.LIMT  =       1               ; DUMMY SDI SEARCH LIMIT
000002                  D.SCHR  =       2               ; DUMMY POINTER TO SUBUNIT CHAR-5
                        ;
                        ;
                        ;       UNIT PARAMETER BITS (U.PARM(R5))
                        ;
100000                  DROP    =       100000          ; DROP BIT (SET IF UNIT OR SUBUNIT DROPPED)
040000                  INITW   =       40000           ; INITIAL WRITE (SET IF INITIAL WRITE IN PROG)
020000                  RESEEK  =       20000           ; IF 1, INDICATES THAT A SEEK IS NECESSARY
010000                  DIREC   =       10000           ; DIRECTION (SET IF SEQUENTIAL ACCESSES DECREASING)
004000                  NEWSUB  =       4000            ; SET IF SEQUENTIAL SEEKS MOVED TO NEW SUBUNIT
002000                  SEKINP  =       2000            ; SEEK IN PROGRESS - SET IF TRUE
001000                  FTIME   =       1000            ; FIRST TIME FLAG - SET FOR INIT CODE
000400                  REVEC   =       400             ; REVECTOR BIT (SET IF BLOCK REVECTORED)
000200                  RBNBN   =       200             ; SEE IF WORKING ON RBN
000100                  REDWRT  =       100             ; REDWRT (READ OR WRITE IN PROGRESS SET IF WRITE)
000040                  REVINP  =       40              ; REVECTORING OPERATION IN PROGRESS
000020                  DATERR  =       20              ; DATA ERROR IF SET
000004                  RETRY   =       4               ; IF CLEAR, START RETRIES AT ZERO
000001                  RCLB    =       1               ; RECALIBRATION BIT (SET IF RECALIBRATE JUST DONE)
                        ;
                        ;       SUBUNIT PARAMETER BITS (S.PARM(R4))
                        ;
020000                  DCYLS   =       20000           ; DIAGNOSTIC CYLINDER FLAG (SET IF DBNS)
010000                  ECCCHK  =       10000           ; 1 IF ECC CORRECTION ALLOWED   MASTER BITS
004000                  RONLY   =       4000            ; READ ONLY (SET IF READ ONLY)
002000                  WONLY   =       2000            ; WRITE ONLY (SET IF WRITE ONLY)
001000                  RTRIES  =       1000            ; 1 IF RETRIES ALLOWED
000200                  ONLYCL  =       200             ; SET IF ONLY CYLINDERS SPECIFIED
                                                        ; USED DURING SETUP ONLY
000100                  SEQSEK  =       100             ; SEQUENTIAL SEEK (START UP TESTING ONLY)
000040                  BEUSED  =       40              ; BEGIN/END SETS (USED IF SET)
000020                  TRACKS  =       20              ; TRACKS OR GROUPS (TRACK IF SET)
000010                  WCHECK  =       10              ; WRITE CHECK BIT (IF SET, WRITE CHECK WILL BE DONE)
                        ;                       WCHECK IS ALSO USED FOR THE UNIT PARAMETERS
000004                  WCHKAL  =       4               ; SET IF WRITE CHECK ALWAYS TO BE DONE
000002                  DATCMP  =       2               ; DATA COMPARE (SET IF DATA COMPARE TO BE DONE)
                        ;                       DATCMP IS ALSO USED FOR THE UNIT PARAMETERS
000001                  DCMPAL  =       1               ; SET IF DATA COMPARE ALWAYS TO BE DONE
```

MACRO DEFINITIONS

```
                        .SBTTL  MACRO DEFINITIONS
                        ;
                        ;       DIAGNOSTIC MACRO FOR TEST4 OVERLAYS
                        ;
                                .MACRO  DIAG$$
                                TST     $$DIAG+$DIAG$
                                BEQ     .+6
                                MOV     #60000,R0
                                MOV     R0,@$$DIAG+$DIAG$
                                BR      .+1
                        $DIAG$  =       $DIAG$ + 1
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;       MESSAGE CONTROL TABLE MACRO
                        ;
                                .MACRO MSG CMDBUF,CMDSZ,RPLBUF,RPLSZ,SUCCOM
                                .WORD CMDBUF                    ;ADDRESS OF COMMAND
                                .WORD CMDSZ                     ;SIZE OF COMMAND IN BYTES
                                .WORD RPLBUF                    ;ADDRESS OF REPLY
                                .WORD RPLSZ                     ;SIZE OF REPLY IN WORDS
                                .IF NB NUMBER
                                .WORD SUCCOM                    ; SUCCESSFUL COMPLETION CODE
                                .ENDC
                                .ENDM
```

MACRO DEFINITIONS

```
      1                                 .MACRO  BCS     LAB..
      2                                                                 BCC     .+2
      3                                                                 BR      LAB..
      4                                 .ENDM
```

MACRO DEFINITIONS

```
                        ;       PUSH REGISTER MACRO
                        ;
                                .MACRO PUSH R9
                                .IRP X,<R9>
                                                                MOV X,-(SP)
                                .ENDR
                                .ENDM
                        ;
                        ;       POP REGISTER MACRO
                        ;
                                .MACRO POP R9
                                .IRP X,<R9>
                                                                MOV (SP)+,X
                                .ENDR
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;ERROR MACROS
                        ;THESE MACROS ARE CALLED TO REPORT ERRORS TO THE HOST PROGRAM.
                        ;THE MACRO NAMES ARE : ERRSF, ERRDF, ERRHRD, ERRSFT. EACH RESULTS IN THE HOST
                        ;BEING REQUESTED TO REPORT THE ERROR.
                        ;ARGUMENTS:     1 (MS$) MESSAGE POINTER
                        ;               2 (P1$) PARAMETER #1
                        ;               3 (P2$) PARAMETER #2
                        ;               4 (P3$) PARAMETER #3
                        ;               5 (P4$) PARAMETER #4
                        ;               6 (P5$) PARAMETER #5
                        ;               7 (P6$) PARAMETER #6
                        ;               8 (P7$) PARAMETER #7
                        ;               9 (P8$) PARAMETER #8
                        ;
                        ;THE MESSAGE POINTER MUST POINT TO AN ADDRESS IN THE OVERLAY 'MS' IMMEDIATELY
                        ;FOLLOWING THE MAIN CODE. ANY ADDRESS MODE MAY BE USED (E.G. #MS1, @R2).
                        ;THE ADDRESS MUST CONTAIN AN ASCII FORMAT STRING TO DETERMINE THE MESSAGE
                        ;TO PRINT.
                        ;THE PARAMETER ARGUMENTS ARE OPTIONAL. THEY SHOULD BE SUPPLIED ONLY WHEN
                        ;THERE IS DATA TO BE PASSED TO THE HOST THAT WILL BE USED IN PRINTING THE
                        ;MESSAGE. THESE PARAMETER ARGUMENTS ARE THE ADDRESS OF DATA TO BE PASSED
                        ;USING ANY ADDRESSING MODE DESIRED.
                        ;ALL REGISTERS ARE RETURNED UNCHANGED. IT SHOULD BE NOTED THAT ARGUMENTS
                        ;CONTAINING SOMETHING OTHER THAN A REGISTER NAME (E.G. #100 OR MEMADR)
                        ;ASSEMBLE TO INSTRUCTIONS THAT SAVE AND RESTORE A REGISTER ON THE STACK.

                                .MACRO ERRSF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ FTLSYS,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRDF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ FTLDEV,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRHRD MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NLIST
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ ERHARD,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .LIST
                                .ENDM

                                .MACRO ERRSFT MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ ERSOFT,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM
```

```
                        ;THE FOLLOWING MACRO ACTUALLY PROCESSES THE ERROR CALL TO THE HOST PROGRAM

                        .MACRO ERROR$ ET$,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,ERRN$
                          .RADIX 8
                          PRMS=ARGS$-1
                          .IF LT,<PRMS>
                            .ERROR;NOT ENOUGH ARGUMENTS IN ERROR CALL
                          .ENDC
                          REGS$=-1
                          .IIF GE,<PRMS-8.>,PARGS. P8$
                          .IIF GE,<PRMS-7.>,PARGS. P7$
                          .IIF GE,<PRMS-6.>,PARGS. P6$
                          .IIF GE,<PRMS-5.>,PARGS. P5$
                          .IIF GE,<PRMS-4.>,PARGS. P4$
                          .IIF GE,<PRMS-3.>,PARGS. P3$
                          .IIF GE,<PRMS-2.>,PARGS. P2$
                          .IIF GE,<PRMS-1.>,PARGS. P1$
                          .IF GE REGS$
                            RSTR$ \REGS$
                          .ENDC
                          .RADIX 10
                          .LIST
                                                                        CALL RERROR     ;ERROR # ERRN$'.
                          .NLIST
                          .RADIX 8
                          .LIST
                                                                        .WORD ET$+ERRN
                                                                        .WORD <PRMS*10000>+MS$
                            .NLIST
                            ERRN=ERRN+1
                        .ENDM

                        .MACRO PARGS.,ADDR$
                          .NTYPE PTYPE$,ADDR$
                          .IF EQ,<PTYPE$&70>
                            .IIF EQ,<PTYPE$&7>-REGS$,RSTR$ \REGS$
                            .LIST
                                                                        MOV ADDR$,-(SP)
                            .NLIST
                          .IFF
                            .IF EQ,<PTYPE$&7>-1          ;PICK A REGISTER TO USE
                              REGU$=2                    ;SELECT R2 IF R1 IS USED IN PARAMETER FETCH
                            .IFF
                              REGU$=1                    ;OTHERWISE USE R1
                            .ENDC
                            .IF NE,<REGU$-REGS$>         ;IF REGISTER NOT ALREADY SAVED
                              .IF GE,REGS$
                                RSTR$ \REGS$             ;RESTORE CURRENT SAVED REGISTER
                              .ENDC
                              SAVR$ \REGU$               ;THEN SAVE SELECTED REGISTER
                            .ENDC
                            GETP$ \REGS$,ADDR$
                          .ENDC
                        .ENDM
```

MACRO DEFINITIONS

```
                        .MACRO SAVR$ REGN
                          .LIST
                                                        MOV R'REGN,SAVREG
                          .NLIST
                          REGS$=REGN
                        .ENDM

                        .MACRO RSTR$ REGN
                          .LIST
                                                        MOV SAVREG,R'REGN
                          .NLIST
                          REGS$=-1
                        .ENDM

                        .MACRO GETP$ REGN,ADDR$
                          .LIST
                                                        MOV ADDR$,R'REGN
                                                        MOV R'REGN,-(SP)
                          .NLIST
                        .ENDM
```

```
                        ;
                        ;       PRIMARY ERROR REPORTING (TEST 4)
                        ;
                        .MACRO  SOFTER  NUM,ARGS
                        ERROR   #ERSOFT,NUM,<ARGS>
                                                        MOV     #ERRMES,R2
                                                        MOV     R2,OUT.RQ

                        .ENDM

                        .MACRO  REPSFT  SFTFLG,ECCFG
                        .IF     NB,SFTFLG
                                                        MOV     #1,OUT.02
                        .IFF
                                                        CLR     OUT.02
                        .ENDC
                        .IF     NB,ECCFG
                                                        MOV     #1,OUT.03
                        .IFF
                                                        CLR     OUT.03
                        .ENDC
                                                        PUSH    R0
                                                        MOV     #U.SNUM,R0
                                                        ADD     R5,R0
                                                        ADD     U.SUBU(R5),R0
                                                        MOV     (R0),OUT.01
                                                        MOV     #T4SOFT,R0
                                                        CALL    HOSTRQ
                                                        POP     R0

                        .ENDM

                        .MACRO  HARDER  NUM,ARGS
                        ERROR   #ERHARD,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ

                        .ENDM

                        .MACRO  DEVFTL  NUM,ARGS
                        ERROR   #FTLDEV,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ

                        .ENDM

                        .MACRO  SYSFTL  NUM,ARGS
                        ERROR   #FTLSYS,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ

                        .ENDM

                        .MACRO  ERROR   TYPE,NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       5
                        MOVMSG  #ER'NUM,4
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
```

```
MACRO DEFINITIONS

                        .ENDC
                                                        MOV     #NUM,OUT.02
                                                        BIS     TYPE,OUT.02
                                                        MOV     #.,OUT.01
                        .RADIX
                        .ENDM

                        .MACRO  CERROR  STNUM,ARGS
                        .RADIX  10
NUMPTR  =               STNUM
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
NUMPTR  =               NUMPTR + 1
                        .ENDR
                        .RADIX
                        .ENDM

                        .MACRO  ERRORC  ARGS
                        .RADIX  10
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
NUMPTR  =               NUMPTR + 1
                        .ENDR
                        .RADIX
                        .ENDM

                        .MACRO  MOVMSG  ARG,INDX

                        .IF     LT,INDX-10
                                                        MOV     ARG,OUT.0'INDX
                        .IFF
                                                        MOV     ARG,OUT.'INDX
                        .ENDC
                        .ENDM

                        .MACRO  ENDERR  POS
                        .IF     NB,POS
                        NDERR   POS
                        .IFF
                        NDERR   \NUMPTR
                        .ENDC
                        .ENDM

                        .MACRO  NDERR   POS
                        .IF     NE,POS
                                                        BIS     #POS,ERRPOS     ; SET THE POSITION
                        .IFF
                                                        CLR     ERRPOS          ; CLEAR THE POSITION
                        .ENDC
                        .ENDM
                ;
                ;       MESSAGE REPORTING MACRO
                ;
                        .MACRO  MSSG    NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       3
                        MOVMSG  #MS'NUM,2
```

MACRO DEFINITIONS

```
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC

                                                        PUSH    <R0,R1>
                                                        MOV     #U.SNUM,R1
                                                        ADD     R5,R1
                                                        ADD     U.SUBU(R5),R1
                                                        MOV     (R1),OUT.01
                                                        MOV     #MESSAG,R0
                                                        CALL    HOSTRQ
                                                        POP     <R1,R0>

                        .RADIX
                        .ENDM
                    ;
                    ;
                        .MACRO  MSSGE   NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       3
                        MOVMSG  #'NUM,2
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC

                                                        PUSH    <R0,R1>
                                                        MOV     #MESSAG,R0
                                                        CALL    HOSTRQ
                                                        POP     <R1,R0>

                        .RADIX
                        .ENDM
```

MACRO DEFINITIONS

```
      1                         ;RETURN DRIVE STATUS MACRO WITH ERROR REPORTING
      2
      3                                 .MACRO  DSTAT,LAB$,E1,E2
      4                                 .NLIST
      5                                 .NLIST  MEB
      6                                 .LIST   ME
      7                                 .LIST
      8                                 CALL    RDSTAT                  ; GET DRIVE STATUS
      9                                 BIT     #10000,R1               ; SEE IF ANY ERRORS
     10                                 BEQ     2$                      ; IF NO ERROR, BRANCH
     11                                 BIT     #4000,R1                ; SEE IF XMIT ERROR
     12                                 BEQ     1$                      ; IF SO, BRANCH
     13                                 ERRHRD  E1                      ; REPORT INVALID STATUS ERROR
     14                                 BR      LAB$                    ; BRANCH TO DONE
     15                         1$:     ERRHRD  E2                      ; REPORT XMIT ERROR
     16                                 BR      LAB$                    ; BRANCH TO DONE
     17                         2$:
     18                                 .NLIST
     19                                 .NLIST  ME
     20                                 .LIST   MEB
     21                                 .LIST
     22                                 .ENDM
```

MACRO DEFINITIONS

```
                        ;
                        ;       XOR THE CONTENTS OF TWO REGISTERS
                        ;
                        .MACRO  RXOR    REG1,REG2
                        MOV     REG2,-(SP)              ; SAVE REGISTER REG2
                        BIC     REG1,REG2               ; CLEAR COORESPONDING BITS IN REG2
                        BIC     (SP)+,REG1              ; CLEAR COORESPONDING BITS IN REG1
                        BIS     REG1,REG2               ; OR WHAT'S LEFT
                        .ENDM
```

MACRO DEFINITIONS

```
                        ;       SDI INTERCHANGE WITH DRIVE WITH ERROR REPORTING
                        ;
                                .MACRO  TALKX   ERRLAB,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   ME
                                .LIST
                                CALL    TALKER                  ; INITIATE SDI INTERCHANGE
                                TST     R3                      ; SEE IF ERROR OCCURRED
                                BEQ     12$                     ; IF NOT, BRANCH
                                BPL     11$                     ; IF SO, BRANCH
                                ERRHRD  E1;SEND COMMAND ERROR
                                BR      ERRLAB
                        11$:    ERRHRD  E2;RECEIVE COMMAND ERROR
                                BR      ERRLAB
                        12$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

START OF TEST CODE

```
                                .SBTTL  START OF TEST CODE
                                ;THE FOLLOWING IS FOR DEBUG PURPOSES ONLY. CAN BE NOP OR BREAKPOINT.
                                ;
000714  114007                          CLR     R0                      ;CHANGE TO BREAKPOINT FOR DEBUG
                                ;
                                ;INITIALIZE STACK
                                ;
000715  104206  001373                  MOV #STACK,SP                   ;SET UP STACK POINTER
000717  001374                  BR      START                           ; BRANCH OVER SUPPORT CODE
                                .SBTTL  RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
000720                          RDSTAT:
                                ;
                                ;RETURN DRIVE STATUS
                                ;STATUS RETURNED IN DM REGISTER 1
                                ;
000720                          PUSH    <R3,R0>                         ; SAVE R3 AND R0
000720  100463                                                                  MOV R3,-(SP)
000721  100467                                                                  MOV R0,-(SP)
000722  104203  000003          MOV     #3,R3                           ; ALLOW ONLY 3 ERRORS
000724  060007                  STATLP: XFC     STATUS                  ; GET DRIVE'S STATUS
000725  103201  014000                  BIC     #14000,R1               ; CLEAR ERROR PASSING BITS
000727  102201  000400                  BIT     #XMTERR,R1              ; CHECK XMIT ERRORS
000731  010737                          BEQ     STATOK                  ; IF NO ERRORS, BRANCH
000732  117403                          DEC     R3                      ; DECREMENT TRANSMIT ERROR COUNT
000733  050724                          BNE     STATLP                  ; IF ERROR COUNT INCOMPLETE, BRANCH
000734  104201  010000                  MOV     #10000,R1               ; FLAG AS TRANSMIT ERROR
000736  000746                          BR      STATEX                  ; BRANCH
000737  102201  000004          STATOK: BIT     #RCVERR,R1              ; RECIEVER ERRORS
000741  050746                          BNE     STATEX                  ; IF VALID, BRANCH
000742  117403                          DEC     R3                      ; DECREMENT ERROR COUNT
000743  050724                          BNE     STATLP                  ; IF ERROR COUNT NON-ZERO, BRANCH
000744  104201  014000                  MOV     #14000,R1               ; FLAG AS INVALID STATUS ERROR
000746                          STATEX: POP     <R0,R3>                 ; RESTORE R0, R3
000746  104267                                                                  MOV (SP)+,R0
000747  104263                                                                  MOV (SP)+,R3
000750  000000                          RETURN                          ; RETURN TO CALLING MODULE
```

```
HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED,

                                        .SBTTL  HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
000751                                  HOSTRQ:
                                        ;
                                        ;SEND REQUEST BUFFER TO HOST AND WAIT FOR RESPONSE.
                                        ;CLEAR ARGUMENT AREA OF OUT BUFFER IN PREPARATION
                                        ;FOR NEXT HOSTRQ CALL.
                                        ;
                                        ;INPUTS:
                                        ;       R0 - HOST REQUEST NUMBER
                                        ;       OUT BUFFER LOADED WITH DATA

000751                                          PUSH <R0,R1,R2>
000751  100467                                                                         MOV R0,-(SP)
000752  100461                                                                         MOV R1,-(SP)
000753  100462                                                                         MOV R2,-(SP)
000754  104070  001006                         MOV R0,OUT.RQ                   ;STORE REQUEST NUMBER IN BUFFER
000756  104207  001006                 SNDAGN: MOV #OUT.RQ,R0                  ;SEND BUFFER TO HOST
000760  104201  000043                         MOV #BUFSIZ,R1
000762  060016                                 XFC MRD
000763  115001                                 TST R1                          ;CHECK FOR ERROR
000764  050756                                 BNE SNDAGN                      ;IF ERROR, TRY AGAIN
000765  104207  001051                         MOV #IN.RQ,R0                   ;WAIT FOR RESPONSE FROM HOST
000767  104201  000043                         MOV #BUFSIZ,R1
000771  060017                                 XFC MWR
000772  104207  001007                         MOV #OUT.01,R0                  ;CLEAR ARGUMENT WORDS IN BUFFER
000774  104201  000034                         MOV #OUT.29-OUT.01,R1
000776  114002                                 CLR R2
000777  100272                         CLRBUF: MOV R2,(R0)+
001000  117401                                 DEC R1
001001  030777                                 BPL CLRBUF
001002                                         POP <R2,R1,R0>
001002  104262                                                                         MOV (SP)+,R2
001003  104261                                                                         MOV (SP)+,R1
001004  104267                                                                         MOV (SP)+,R0
001005  000000                                 RETURN
```

```
                                ;STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS
                                ;OUT BUFFER - DATA TO SEND TO HOST
001006  000000                  OUT.RQ: .WORD 0                 ;HOST REQUEST CODE
001007  000000                  OUT.01: .WORD 0                 ;DATA ARGUMENT  1
001010  000000                  OUT.02: .WORD 0                 ;DATA ARGUMENT  2
001011  000000                  OUT.03: .WORD 0                 ;DATA ARGUMENT  3
001012  000000                  OUT.04: .WORD 0                 ;DATA ARGUMENT  4
001013  000000                  OUT.05: .WORD 0                 ;DATA ARGUMENT  5
001014  000000                  OUT.06: .WORD 0                 ;DATA ARGUMENT  6
001015  000000                  OUT.07: .WORD 0                 ;DATA ARGUMENT  7
001016  000000                  OUT.08: .WORD 0                 ;DATA ARGUMENT  8
001017  000000                  OUT.09: .WORD 0                 ;DATA ARGUMENT  9
001020  000000                  OUT.10: .WORD 0                 ;DATA ARGUMENT 10
001021  000000                  OUT.11: .WORD 0                 ;DATA ARGUMENT 11
001022  000000                  OUT.12: .WORD 0                 ;DATA ARGUMENT 12
001023  000000                  OUT.13: .WORD 0                 ;DATA ARGUMENT 13
001024  000000                  OUT.14: .WORD 0                 ;DATA ARGUMENT 14
001025  000000                  OUT.15: .WORD 0                 ;DATA ARGUMENT 15
001026  000000                  OUT.16: .WORD 0                 ;DATA ARGUMENT 16
001027  000000                  OUT.17: .WORD 0                 ;DATA ARGUMENT 17
001030  000000                  OUT.18: .WORD 0                 ;DATA ARGUMENT 18
001031  000000                  OUT.19: .WORD 0                 ;DATA ARGUMENT 19
001032  000000                  OUT.20: .WORD 0                 ;DATA ARGUMENT 20
001033  000000                  OUT.21: .WORD 0                 ;DATA ARGUMENT 21
001034  000000                  OUT.22: .WORD 0                 ;DATA ARGUMENT 22
001035  000000                  OUT.23: .WORD 0                 ;DATA ARGUMENT 23
001036  000000                  OUT.24: .WORD 0                 ;DATA ARGUMENT 24
001037  000000                  OUT.25: .WORD 0                 ;DATA ARGUMENT 25
001040  000000                  OUT.26: .WORD 0                 ;DATA ARGUMENT 26
001041  000000                  OUT.27: .WORD 0                 ;DATA ARGUMENT 27
001042  000000                  OUT.28: .WORD 0                 ;DATA ARGUMENT 28
001043  000000                  OUT.29: .WORD 0                 ;DATA ARGUMENT 29
001044  000000                  OUT.30: .WORD 0                 ;DATA ARGUMENT 30
001045  000000                  OUT.31: .WORD 0                 ;DATA ARGUMENT 31
001046  000000                  OUT.32: .WORD 0                 ;DATA ARGUMENT 32
001047  000000                  OUT.33: .WORD 0                 ;DATA ARGUMENT 33
001050  000000                  OUT.34: .WORD 0                 ;DATA ARGUMENT 34

                                ;IN BUFFER - DATA RECEIVED FROM HOST

001051  000000                  IN.RQ:  .WORD 0                 ;HOST REQUEST CODE (ECHO)
001052  000000                  IN.01:  .WORD 0                 ;DATA ARGUMENT  1
001053  000000                  IN.02:  .WORD 0                 ;DATA ARGUMENT  2
001054  000000                  IN.03:  .WORD 0                 ;DATA ARGUMENT  3
001055  000000                  IN.04:  .WORD 0                 ;DATA ARGUMENT  4
001056  000000                  IN.05:  .WORD 0                 ;DATA ARGUMENT  5
001057  000000                  IN.06:  .WORD 0                 ;DATA ARGUMENT  6
001060  000000                  IN.07:  .WORD 0                 ;DATA ARGUMENT  7
001061  000000                  IN.08:  .WORD 0                 ;DATA ARGUMENT  8
001062  000000                  IN.09:  .WORD 0                 ;DATA ARGUMENT  9
001063  000000                  IN.10:  .WORD 0                 ;DATA ARGUMENT 10
001064  000000                  IN.11:  .WORD 0                 ;DATA ARGUMENT 11
001065  000000                  IN.12:  .WORD 0                 ;DATA ARGUMENT 12
001066  000000                  IN.13:  .WORD 0                 ;DATA ARGUMENT 13
001067  000000                  IN.14:  .WORD 0                 ;DATA ARGUMENT 14
```

```
     58 001070  000000                IN.15:  .WORD 0                 ;DATA ARGUMENT 15
     59 001071  000000                IN.16:  .WORD 0                 ;DATA ARGUMENT 16
     60 001072  000000                IN.17:  .WORD 0                 ;DATA ARGUMENT 17
     61 001073  000000                IN.18:  .WORD 0                 ;DATA ARGUMENT 18
     62 001074  000000                IN.19:  .WORD 0                 ;DATA ARGUMENT 19
     63 001075  000000                IN.20:  .WORD 0                 ;DATA ARGUMENT 20
     64 001076  000000                IN.21:  .WORD 0                 ;DATA ARGUMENT 21
     65 001077  000000                IN.22:  .WORD 0                 ;DATA ARGUMENT 22
     66 001100  000000                IN.23:  .WORD 0                 ;DATA ARGUMENT 23
     67 001101  000000                IN.24:  .WORD 0                 ;DATA ARGUMENT 24
     68 001102  000000                IN.25:  .WORD 0                 ;DATA ARGUMENT 25
     69 001103  000000                IN.26:  .WORD 0                 ;DATA ARGUMENT 26
     70 001104  000000                IN.27:  .WORD 0                 ;DATA ARGUMENT 27
     71 001105  000000                IN.28:  .WORD 0                 ;DATA ARGUMENT 28
     72 001106  000000                IN.29:  .WORD 0                 ;DATA ARGUMENT 29
     73 001107  000000                IN.30:  .WORD 0                 ;DATA ARGUMENT 30
     74 001110  000000                IN.31:  .WORD 0                 ;DATA ARGUMENT 31
     75 001111  000000                IN.32:  .WORD 0                 ;DATA ARGUMENT 32
     76 001112  000000                IN.33:  .WORD 0                 ;DATA ARGUMENT 33
     77 001113  000000                IN.34:  .WORD 0                 ;DATA ARGUMENT 34
     78         000043                BUFSIZ  =       . - IN.RQ       ;SIZE OF BUFFER
```

```
                        ;
                        ;       TALKER SENDS AND RECEIVES AN SDI INTERCHANGE
                        ;
                        ;INPUTS:
                        ;       R2 - SDI INTERCONNECT
                        ;       R3 - POINTER TO COMMAND TABLE CONTAINING APPROPRIATE COMMAND
                        ;       SDILTO/SDISTO SDI LONG AND SHORT TIMEOUTS, RESPECTIVELY
                        ;OUTPUTS:
                        ;       R0 - RETURN OP CODE FROM UNIT
                        ;       R3 - ERROR CODE - 0 = NO ERROR, 1 = RECEIVE ERROR, 100001 = SEND ERROR
                        ;
001114                  TALKER: PUSH    <R1,R4>         ; SAVE REGISTERS
001114  100461                                                          MOV R1,-(SP)
001115  100464                                                          MOV R4,-(SP)
001116  104237                  MOV     (R3)+,R0        ; SET ADR OF SDI COMMAND BUFFER
001117  104231                  MOV     (R3)+,R1        ; SET BUFFER LENGTH
001120  060004                  XFC     SEND            ; SEND COMMAND
001121  115001                  TST     R1              ; DID UNIT ACCEPT COMMAND
001122  011126                  BEQ     TALK1A          ; IF SO, BRANCH
001123  104203  100001          MOV     #100001,R3      ; FLAG AS SEND ERROR
001125  001157                  BR      TALK2B          ; BRANCH TO EXIT
001126  106203  001374  TALK1A: CMP     #LONG,R3        ; SEE IF LONG TIMEOUT TO BE USED
001130  071134                  BMI     1$              ; IF SO, BRANCH
001131  104304  001162          MOV     SDISTO,R4       ; SET UP SHORT TIMEOUT
001133  001136                  BR      TALK1B          ; BRANCH
001134  104304  0C1163  1$:     MOV     SDILTO,R4       ; SET UP LONG TIMEOUT
001136  104137          TALK1B: MOV     (R3),R0         ; SET DATA BUFFER ADDRESS
001137  104631  000001          MOV     1(R3),R1        ; SET BUFFER LENGTH
001141  060005                  XFC     RCV             ; SEND RECEIVE SDI COMMAND
001142  115001                  TST     R1              ; DID ERROR OCCUR
001143  011156                  BEQ     TALK2A          ; IF NOT, BRANCH
001144  106201  000001          CMP     #1,R1           ; SEE IF TIMEOUT
001146  011151                  BEQ     1$              ; IF SO, BRANCH
001147  104013                  MOV     R1,R3           ; MOVE ERROR TYPE TO R3 FOR REPORTING
001150  001157                  BR      TALK2B          ; EXIT
001151  117404          1$:     DEC     R4              ; DECREMENT TIMEOUT VALUE
001152  051136                  BNE     TALK1B          ; IF NOT TIMEOUT, BRANCH
001153  104203  000001          MOV     #1,R3           ; FLAG AS RECIEVE ERROR
001155  001157                  BR      TALK2B          ; BRANCH TO EXIT
001156  114003          TALK2A: CLR     R3              ; FLAG AS NO ERRORS
001157                  TALK2B: POP     <R4,R1>         ; RESTORE R4, R1
001157  104264                                                          MOV (SP)+,R4
001160  104261                                                          MOV (SP)+,R1
001161  000000                  RETURN
                        ;
001162  000012          SDISTO: .WORD   10.             ; SDI SHORT TIMEOUT
001163  000024          SDILTO: .WORD   20.             ; SDI LONG TIMEOUT
```

```
      1                                 ;MREAD
      2                                 ;
      3                                 ;READ ONE WORD FROM UNIBUS MEMORY
      4                                 ;INPUTS:
      5                                 ;       R5 - ADDRESS OF WORD TO READ (LOW 16 BITS)
      6                                 ;       R4 -    ''           (HIGH 2 BITS)
      7                                 ;OUTPUTS:
      8                                 ;       R0 - DATA READ
      9
     10 001164                          MREAD:  PUSH <R2,R3>                    ;SAVE SOME REGISTERS
        001164  100462                                                                  MOV R2,-(SP)
        001165  100463                                                                  MOV R3,-(SP)
     11 001166  104057                          MOV R5,R0                       ;PUT UNIBUS ADDRESS IN R0 AND R1
     12 001167  104041                          MOV R4,R1
     13 001170  104202  000001               MOV #1,R2                       ;TRANSFER ONE WORD
     14 001172  104203  001222               MOV #UDATA,R3                   ;LOCATION TO PUT DATA
     15 001174  060013                       XFC UREAD                       ;DO THE READ
     16 001175  104307  001222               MOV UDATA,R0                    ;GET DATA READ
     17 001177                          MOV <R3,R2>                     ;RESTORE OTHER REGISTERS
        001177  104263                                                                  MOV (SP)+,R3
        001200  104262                                                                  MOV (SP)+,R2
     18 001201  000000                          RETURN
```

```
                                        ;MWRITE
                                        ;
                                        ;WRITE ONE WORD TO UNIBUS MEMORY
                                        ;INPUTS:
                                        ;       R5 - ADDRESS OF WORD TO WRITE (LOW 16 BITS)
                                        ;       R4 -    "            (HIGH 2 BITS)
                                        ;       R0 - DATA TO WRITE
                                        ;OUTPUTS:

001202                                  MWRITE: PUSH <R0,R2,R3>                 ;SAVE SOME REGISTERS
001202  100467                                                                         MOV R0,-(SP)
001203  100462                                                                         MOV R2,-(SP)
001204  100463                                                                         MOV R3,-(SP)
001205  104070  001222                          MOV R0,UDATA                    ;PUT DATA TO BE WRITTEN INTO BUFFER
001207  104057                                  MOV R5,R0                       ;PUT UNIBUS ADDRESS IN R0 AND R1
001210  104041                                  MOV R4,R1
001211  104202  000001                          MOV #1,R2                       ;TRANSFER ONE WORD
001213  104203  001222                          MOV #UDATA,R3                   ;ADDRESS OF DATA WORD
001215  060014                                  XFC UWRITE                      ;DO THE WRITE
001216                                          POP <R3,R2,R0>                  ;RESTORE THE REGISTERS
001216  104263                                                                         MOV (SP)+,R3
001217  104262                                                                         MOV (SP)+,R2
001220  104267                                                                         MOV (SP)+,R0
001221  000000                                  RETURN

001222  000000                          UDATA:  .WORD 0                         ;DATA BUFFER FOR TRANSFERS TO AND FROM
                                                                                ;UNIBUS MEMORY
```

```
                                ;XOR
                                ;
                                ;PERFORM XOR LOGIC FUNCTION ON TWO REGISTERS
                                ;INPUTS:
                                ;       R1, R2 - DATA TO BE XOR'ED
                                ;OUTPUTS:
                                ;       R1 - UNCHANGED
                                ;       R2 - XOR OF TWO INPUTS
                                ;
001223                  XOR:    PUSH R3                         ;SAVE R3
001223  100463                                                                  MOV R3,-(SP)
001224  104013                  MOV R1,R3
001225  103023                  BIC R2,R3
001226  103012                  BIC R1,R2
001227  101032                  BIS R3,R2
001230                          POP R3                          ;RESTORE R3
001230  104263                                                                  MOV (SP)+,R3
001231  115002                  TST R2                          ;SET CONDITION CODES
001232  000000                  RETURN
```

```
                                TO:
                                ;
                                ;       CALCULATE THE TIMEOUT IN 9SEC INTERVALS (SDI RECEIVE XFC TAKES
                                ;       9 SEC)
                                ;
001233  104201  000001                MOV     #1,R1           ; SET UP LOG2 SHIFTER
001235  110201                1$:     ROL     R1              ; DOUBLE THE TIMEOUT VALUE
001236  103201  000001                BIC     #1,R1           ; CLEAR THE LOW BIT
001240  117407                        DEC     R0              ; DECREMENT COUNT
001241  051235                        BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
001242  114007                        CLR     R0              ; CLEAR 9SEC COUNT
001243  115407                2$:     INC     R0              ; INCREMENT 9 SEC COUNT
001244  107201  000011                SUB     #9.,R1          ; SUBTRACT 9 SEC FROM TIMEOUT
001246  031243                        BPL     2$              ; IF MORE TIME TO GO, BRANCH
001247  000000                        RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                ;RERROR
                                ;
                                ;REPORT ERROR TO HOST PROGRAM
                                ;THIS ROUTINE IS CALLED BY THE ERROR MACROS:
                                ;ERRSF, ERRDF, ERRHRD AND ERRSFT
001250                          RERROR:
001250                                  PUSH R0                         ;SAVE ONE REGISTER
001250  100467                                                                  MOV R0,-(SP)
001251  104067                          MOV SP,R0                       ;GET STACK POINTER
001252                                  PUSH <R1,R2,R3,R4>              ;SAVE MORE REGISTERS
001252  100461                                                                  MOV R1,-(SP)
001253  100462                                                                  MOV R2,-(SP)
001254  100463                                                                  MOV R3,-(SP)
001255  100464                                                                  MOV R4,-(SP)
001256  115407                          INC R0                          ;CHANGE SAVED STACK POINTER TO POINT TO
                                                                        ; ADDRESS OF LOCATION AFTER CALL
001257  104271                          MOV (R0)+,R1                    ;GET RETURN PC
001260  104202  001007                  MOV #OUT.01,R2                  ;GET ADDRESS OF WHERE TO PUT DATA
001262  117401                          DEC R1                          ;REDUCE TO PC OF ERROR CALL
001263  100221                          MOV R1,(R2)+                    ;PUT PC IN OUT BUFFER
001264  115401                          INC R1                          ;GET BACK TO RETURN PC
001265  104213                          MOV (R1)+,R3                    ;GET ERROR NUMBER AND TYPE
001266  100223                          MOV R3,(R2)+                    ;PUT IN BUFFER
001267  104303  001331                  MOV LUNIT,R3                    ;PUT UNIT NUMBER IN BUFFER
001271  100223                          MOV R3,(R2)+
001272  104113                          MOV (R1),R3                     ;GET MESSAGE POINTER
001273  103203  170000                  BIC #^C007777,R3                ;CLEAR OTHER BITS
001275  100223                          MOV R3,(R2)+                    ;PUT IN OUT BUFFER
001276  104214                          MOV (R1)+,R4                    ;GET COUNT OF PARAMETERS
                                                                        ;R1 IS NOW POINTING TO INSTRUCTION AFTER ERROR CALL
001277  110704                          SWAB R4
001300  110604                          ROR R4                          ;EXTRACT NUMBER OF PARAMETERS FROM ERROR MACRO
001301  110604                          ROR R4
001302  110604                          ROR R4
001303  110604                          ROR R4
001304  103204  177760                  BIC #177760,R4
001306  104040  001327                  MOV R4,SPADJU+1                 ;SAVE FOR LATER ADJUSTMENT OF STACK POINTER
001310  011315                          BEQ RERRCA                      ;BRANCH IF NO PARAMETERS
001311  104273                  RERRPA: MOV (R0)+,R3                    ;GET PARAMETER
001312  100223                          MOV R3,(R2)+                    ;STORE PARAMETER IN OUT BUFFER
001313  117404                          DEC R4                          ;COUNT THE PARAMETERS
001314  051311                          BNE RERRPA                      ;GET NEXT IF MORE
001315  100471                  RERRCA: MOV R1,-(R0)                    ;PUT RETURN ADDRESS ON STACK
001316  104207  060013                  MOV #ERRMES,R0                  ;SEND ERROR PACKET TO HOST PROGRAM
001320  020751                          CALL HOSTRQ
001321                                  POP <R4,R3,R2,R1,R0>            ;RESTORE REGISTERS
001321  104264                                                                  MOV (SP)+,R4
001322  104263                                                                  MOV (SP)+,R3
001323  104262                                                                  MOV (SP)+,R2
001324  104261                                                                  MOV (SP)+,R1
001325  104267                                                                  MOV (SP)+,R0
001326  105206  000000          SPADJU: ADD #0,SP                       ;ADJUST STACK OVER PARAMETERS
                                                                        ;VALUE CHANGED ABOVE
001330  000000                          RETURN

001331  177777                  LUNIT:  .WORD -1                        ;LOGICAL UNIT NUMBER (-1 FOR NOT AVAILABLE)
```

```
     48 001332  000000                SAVREG: .WORD 0                 ;STORAGE FOR REGISTER AT CALL TIME
```

```
                                ;STACK AREA
                                
001333  123456                          .WORD 123456            ;END MARKER FOR STACK
001334                                  .BLKW 31.               ;STACK
001373  123456                  STACK:  .WORD 123456            ;MARKER FOR STACK UNDERFLOW
```

```
                                .SBTTL  FREE MEMORY CHECK

                        001750          ERRN=1000.                      ;START ERROR NUMBERS FROM 1000.

                                ;ASK HOST WHERE FREE MEMORY IS AND TO FILL IT WITH AN ADDRESS PATTERN

001374                          LONG:                                   ;UNUSED LABEL THAT MUST BE DEFINED
001374  104207  060000          START:  MOV #T1MSIZ,R0                  ;GET REQUEST NUMBER
001376  020751                          CALL HOSTRQ                     ;ASK HOST
001377  104207  001052                  MOV #IN.01,R0                   ;TRANSFER DATA FROM HOST
001401  104201  002416                  MOV #FWADR,R1                   ; TO STORAGE
001403  104202  000010                  MOV #8.,R2
001405  104273          FMEMFL: MOV (R0)+,R3
001406  100213                  MOV R3,(R1)+
001407  117402                  DEC R2
001410  051405                  BNE FMEMFL

                                ;READ ALL OF SPECIFIED MEMORY AND CHECK FOR DATA SAME AS ADDRESS

001411  104305  002416                  MOV FWADR,R5                    ;GET STARTING ADDRESS
001413  104304  002417                  MOV FWADRH,R4
001415  021164          ACHK1:  CALL MREAD                              ;READ FROM UNIBUS MEMORY
001416  115001                  TST     R1                              ;WAS IT OK?
001417  011440                  BEQ     ACHK2                           ;IF SO, CONTINUE
001420  117401                  DEC     R1                              ;IF R1 = 1, NON EXITENT MEMORY
001421  051430                  BNE     1$                              ; IF NOT, CONTINUE
001422                          ERRHRD  MS1000,R5,R4
001422  100464                                                                  MOV R4,-(SP)
001423  100465                                                                  MOV R5,-(SP)
001424  021250                                                                  CALL RERROR     ;ERROR # 1000.
001425  101750                                                                  .WORD ERHARD+ERRN
001426  020000                                                                  .WORD <PRMS*10000>+MS1000
001427  001451                  BR      ACHK3                           ;CONTINUE
001430                  1$:     ERRHRD  MS1001,R5,R4,R5,R0
001430  100467                                                                  MOV R0,-(SP)
001431  100465                                                                  MOV R5,-(SP)
001432  100464                                                                  MOV R4,-(SP)
001433  100465                                                                  MOV R5,-(SP)
001434  021250                                                                  CALL RERROR     ;ERROR # 1001.
001435  101751                                                                  .WORD ERHARD+ERRN
001436  040063                                                                  .WORD <PRMS*10000>+MS1001
001437  001451                  BR      ACHK3                           ;CONTINUE

                                ;COMPARE DATA READ WITH EXPECTED

001440  106075          ACHK2:  CMP R0,R5                               ;COMPARE DATA READ WITH ADDRESS
001441  011451                  BEQ ACHK3                               ;BRANCH IF A MATCH
001442                          ERRHRD MS1002,R5,R4,R5,R0 ;UNIBUS ADDRESSING ERROR - INCORRECT DATA READ
001442  100467                                                                  MOV R0,-(SP)
001443  100465                                                                  MOV R5,-(SP)
001444  100464                                                                  MOV R4,-(SP)
001445  100465                                                                  MOV R5,-(SP)
001446  021250                                                                  CALL RERROR     ;ERROR # 1002.
001447  101752                                                                  .WORD ERHARD+ERRN
001450  040170                                                                  .WORD <PRMS*10000>+MS1002

                                ;INCREMENT TO NEXT LOCATION UNTIL ALL TESTED
```

FREE MEMORY CHECK

```
     39
     40 001451  106305  002420          ACHK3:  CMP LWADR,R5                    ;CHECK IF AT LAST ADDRESS
     41 001453  051457                          BNE ACHK4
     42 001454  106304  002421                  CMP LWADRH,R4
     43 001456  011464                          BEQ BCHK                        ;GO TO NEXT TEST IF SO
     44 001457  105205  000002          ACHK4:  ADD #2,R5                       ;INCREMENT TEST ADDRESS
     45 001461  041415                          BCC ACHK1
     46 001462  115404                          INC R4                          ;CARRY TO HIGH BITS
     47 001463  001415                          BR ACHK1                        ;LOOP IF STILL IN SPECIFIED MEMORY
```

```
                                ;CHECK EACH ADDRESS LINE BY LOOKING AT TWO LOCATIONS WITH ONLY THAT
                                ;ADDRESS LINE DIFFERENT AND VERIFYING TWO LOCATIONS ARE ACTUALLY ACCESSED
                                
001464  104307  002417    BCHK:   MOV FWADRH,R0           ;LOAD TEST ADDRESS HIGH BITS
001466  104070  002427            MOV R0,TADRH
001470  104203  000002            MOV #2,R3               ;GET STARTING ADDRESS BIT TO TEST
001472  104032            BCHK1:  MOV R3,R2               ;GET ADDRESS BIT TO TEST
001473  104301  002416            MOV FWADR,R1            ;GET FIRST ADDRESS OF FREE MEMORY
001475  021223                    CALL XOR                ;CHANGE JUST THE ONE BIT IN ADDRESS
001476  104020  002426            MOV R2,TADR             ;STORE TEST ADDRESS
001500  022273                    CALL BCHKM              ;TEST THE MEMORY ADDRESS
001501  105033                    ADD R3,R3               ;GET NEW ADDRESS BIT TO TEST
001502  051472                    BNE BCHK1               ;LOOP IF STILL AN ADDRESS BIT TO TEST
                                
001503  104307  002416            MOV FWADR,R0            ;NOW MOVE TO HIGH TWO BITS
001505  104070  002426            MOV R0,TADR             ;COPY LOW BITS TO TEST ADDRESS
001507  104203  000001            MOV #1,R3               ;START BIT
001511  104032            BCHK2:  MOV R3,R2               ;GET TEST BIT
001512  104301  002417            MOV FWADRH,R1           ;GET FIRST ADDRESS
001514  021223                    CALL XOR                ;CHANGE ONLY THE ONE BIT
001515  104020  002427            MOV R2,TADRH            ;STORE AS TEST ADDRESS
001517  022273                    CALL BCHKM              ;TEST THE MEMORY ADDRESS
001520  105033                    ADD R3,R3               ;CHANGE TEST BIT
001521  102203  000004            BIT #4,R3               ;CHECK IF PAST TWO ADDRESS BITS
001523  011511                    BEQ BCHK2               ;REPEAT FOR OTHER BIT
```

```
                                ;TRANSFER LARGE BUFFERS OF DATA TO AND FROM THE HOST MEMORY.
                                ;
                                ;THE HOST MEMORY WILL BE DIVIDED INTO SEVERAL BUFFERS. ALL BUFFERS WILL
                                ;BE WRITTEN WITH PATTERN 0 THEN READ AND THE DATA COMPARED TO PATTERN 0.
                                ;EACH BUFFER WILL THEN BE WRITTEN TO PATTERN 1 WITH A READ OF ALL BUFFERS
                                ;AFTER EACH WRITE. WHEN ALL BUFFERS HAVE BEEN WRITTEN WITH PATTERN 1, THEN
                                ;THE SEQUENCE REPEATS BY WRITING EACH BUFFER WITH PATTERN 2 AND THEN
                                ;WITH PATTERN 3.
                                ;
                                ;DATA PATTERNS: (EACH IS A REPETITION OF THREE WORDS)
                                ;0 - 111111     1 - 177400      2 - 155555      3 - 000377
                                ;    044444         007760          133333          170017
                                ;    022222         000377          066666          177400

                                ;BREAK THE HOST MEMORY INTO BUFFERS

001524  104202  010000        CCHK:   MOV #HIMEM,R2                   ;COMPUTE SIZE OF DATA
001526  107202  002460                SUB #FREE,R2                    ;  BUFFER IN M MEMORY
001530  105022                      ADD R2,R2                       ;CHANGE TO BYTE COUNT

                                ; *** FIND OUT IF WRITABLE HOST SPACE IS LESS THAN AVAILABLE DM SPACE
001531  104304  002420                MOV     LWADR,R4                ;R4 = LAST WRITABLE HOST BUFFER WORD LOCATION
001533  104303  002421                MOV     LWADRH,R3               ;R3 = EXTENDED ADDRESS BITS OF SAME
001535  107304  002416                SUB     FWADR,R4                ;R4 = WRITABLE BUFFER SIZE
001537  041541                      BCC     1$                      ;IF DIDN'T CROSS PAGE BOUNDARY, CONTINUE
001540  117403                      DEC     R3                      ;ELSE, DECREMENT EXTENDED ADDRESS BITS BY 1
001541  107303  002417        1$:     SUB     FWADRH,R3               ;R3 = EXTENDED ADDRESS OF WRITABLE BUFFER SIZE
001543  105203  000000                ADD     #0,R3                   ;CLEAR CARRY

                                ; *** DIVIDE WRITABLE REGION SIZE BY HALF
001545  110603                      ROR     R3                      ;ROTATE EXTENDED ADDRESS BITS INTO CARRY
001546  110604                      ROR     R4                      ;ROTATE EXTENDED ADDRESS IN & GET AREA/2
001547  102204  000001              BIT     #BIT00,R4               ;MAKE SURE THE BYTE COUNT IS EVEN
001551  011553                      BEQ     2$                      ;IF IT IS, CONTINUE
001552  117404                      DEC     R4                      ;ELSE, FORCE BYTE COUNT TO BE EVEN
001553  115003                2$:     TST     R3                      ;IS R3 = 0?
001554  051560                      BNE     3$                      ;IF NOT, WRITABLE AREA >>> AVAILABLE DM BUFFER SPACE
001555  106042                      CMP     R4,R2                   ;IS WRITABLE AREA < AVAILABLE DM BUFFER SPACE
001556  041560                      BCC     3$                      ;IF NOT, CONTINUE
001557  104042                      MOV     R4,R2                   ;ELSE, DM SPACE > HOST WRITABLE SPACE
                                                                    ;  STORE IN R2 & CONTINUE
001560  104205  002460        3$:     MOV #FREE,R5                    ;GET ADDRESS OF FIRST TABLE ENTRY
001562  104304  002416                MOV FWADR,R4                    ;GET ADDRESS OF FIRST
001564  104303  002417                MOV FWADRH,R3                   ;  BUFFER IN HOST
001566  114001                      CLR R1                          ;INIT COUNT OF BUFFERS
001567  100254                4$:     MOV R4,(R5)+                    ;STORE HOST BUFFER ADDRESS
001570  100253                      MOV R3,(R5)+                    ;  IN TABE ENTRY
001571  107202  000004              SUB #4,R2                       ;REDUCE BUFFER SIZE BY TABLE ENTRY
001573  105024                      ADD R2,R4                       ;INCREADE HOST ADDRESS TO
001574  041576                      BCC 5$                          ;  NEXT BUFFER
001575  115403                      INC R3
001576  106303  002421        5$:     CMP LWADRH,R3                   ;CHECK IF BUFFER LARGER
001600  071610                      BMI 7$                          ;  THEN HOST MEMORY REMAINING
001601  051606                      BNE 6$
001602  106304  002420              CMP LWADR,R4
001604  041606                      BCC 6$
001605  001610                      BR 7$
001606  115401                6$:     INC R1                          ;COUNT THE TABLE ENTRY
001607  001567                      BR 4$                           ;GO BACK AND DO AGAIN
```

```
      58
      59 001610  104050  002433          7$:     MOV R5,DATBUF           ;SA.~ ADDRESS OF DATA BUFFER
      60 001612  105207  000000                  ADD #0,R0               ;CLEAR CARRY
      61 001614  110602                          ROR R2                  ;SAVE SIZE OF DATA BUFFER
      62 001615  104020  002434                  MOV R2,SIZBUF           ;  IN WORDS
      63 001617  104010  002430                  MOV R1,BUFCNT           ;SAVE COUNT OF BUFFERS
```

```
                                        ;WRITE ALL BUFFERS TO PATTERN 0

001621  114007                                CLR R0                  ;LOAD PATTERN 0
001622  104070  002432                        MOV R0,CURPAT           ;  IN SAVE WORD
001624  104070  002431          8$:           MOV R0,CURBUF           ;SELECT BUFFER 0
001626  021666                                CALL WRITE              ;WRITE THE BUFFER
001627  104307  002431                        MOV CURBUF,R0           ;INCREMENT TO NEXT BUFFER
001631  115407                                INC R0
001632  106307  002430                        CMP BUFCNT,R0           ;WRITE ANOTHER IF
001634  051624                                BNE 8$                  ;  NOT AT LAST
001635  021742                                CALL READ               ;READ ALL HOST BUFFERS

001636  104307  002432          9$:           MOV CURPAT,R0           ;INCREMENT PATTERN NUMBER
001640  115407                                INC R0
001641  106207  000004                        CMP #4,R0               ;EXIT SUBTEST IF NO
001643  011662                                BEQ DONECD              ;  NO PATTERNS REMAINING
001644  104070  002432                        MOV R0,CURPAT

001646  114007                                CLR R0                  ;POINT TO BUFFER 0
001647  104070  002431          10$:          MOV R0,CURBUF
001651  021666                                CALL WRITE              ;WRITE A BUFFER
001652  021742                                CALL READ               ;READ ALL HOST BUFFERS
001653  104307  002431                        MOV CURBUF,R0           ;INCREMENT TO NEXT BUFFER
001655  115407                                INC R0
001656  106307  002430                        CMP BUFCNT,R0           ;CHECK IF AT LAST
001660  051647                                BNE 10$                 ;  WRITE NEXT BUFFER
001661  001636                                BR 9$                   ;  WRITE NEXT PATTERN
```

```
      1 001662  104207  060016          DONECD: MOV #DONE,R0            ;END OF PROGRAM, TELL PDP-11
      2 001664  020751                          CALL HOSTRQ
      3 001665  001662                          BR DONECD               ;KEEP SENDING IF RETURNED
```

```
                                ;FILL A DATA BUFFER BEGINNING AT ADDRESS IN DATBUF AND WHOSE SIZE
                                ;IS IN SIZBUF WITH A DATA PATTERN SPECIFIED BY CONTENTS OF CURPAT.
                                ;WRITE THIS BUFFER TO HOST STARTING AT ADDRESS IN TWO WORDS POINTED
                                ;TO BY CURBUF. THEN PLACE THE PATTERN NUMBER IN THE SECOND WORD OF THE
                                ;HOST ADDRESS

                                ;FILL BUFFER WITH DATA PATTERN

001666  104307  002432        WRITE:  MOV CURPAT,R0                   ;GET PATTERN NUMBER
001670  105077                        ADD R0,R0                       ;  TIMES FOUR
001671  105077                        ADD R0,R0
001672  104072                        MOV R0,R2                       ;SAVE FOR ADDING TO TABLE ENTRY
001673  105207  002254                ADD #PAT0,R0                    ;ADD START OF PATTERN TABLES
001675  104273                        MOV (R0)+,R3                    ;GET PATTERN WORDS
001676  104274                        MOV (R0)+,R4
001677  104275                        MOV (R0)+,R5
001700  104307  002433                MOV DATBUF,R0                   ;GET ADDRESS OF BUFFER
001702  104301  002434                MOV SIZBUF,R1                   ;GET SIZE OF BUFFER

001704  100273                1$:     MOV R3,(R0)+                    ;LOAD ONE WORD
001705  117401                        DEC R1                          ;COUNT THE WORDS
001706  011715                        BEQ 2$
001707  100274                        MOV R4,(R0)+                    ;LOAD ONE WORD
001710  117401                        DEC R1                          ;COUNT THE WORDS
001711  011715                        BEQ 2$
001712  100275                        MOV R5,(R0)+                    ;LOAD ONE WORD
001713  117401                        DEC R1                          ;COUNT THE WORDS
001714  051704                        BNE 1$

                                ;WRITE THE BUFFER TO HOST MEMORY

001715  104305  002431        2$:     MOV CURBUF,R5                   ;GET POINTER TO HOST ADDRESS
001717  105055                        ADD R5,R5                       ;  COUNT TIMES TWO
001720  105205  002460                ADD #FREE,R5                    ;  PLUS START ADDRESS
001722  104257                        MOV (R5)+,R0                    ;GET LOW ADDRESS BITS
001723  104151                        MOV (R5),R1                     ;GET HIGH ADDRESS BITS
001724  103201  177774                BIC #177774,R1                  ;CLEAR CURRENT PATTERN
001726  104070  002447                MOV     R0,LBUFWL               ;SAVE LAST BUFFER WRITTEN
001730  104010  002450                MOV     R1,LBUFWH
001732  101012                        BIS R1,R2                       ;SET IN NEW PATTERN
001733  100152                        MOV R2,(R5)                     ;STORE IN TABLE ENTRY
001734  104302  002434                MOV SIZBUF,R2                   ;GET WORDS TO TRANSFER
001736  104303  002433                MOV DATBUF,R3                   ;GET DM ADDRESS
001740  060014                        XFC UWRITE                      ;WRITE TO THE HOST MEMORY
001741  000000                        RETURN
```

```
                        ;READ AND PERFORM A DATA COMPARE ON ALL THE HOST BUFFERS. TWO WORD
                        ;TABLE ENTRIES STARTING AT FREE CONTAIN THE HOST ADDRESS OF THE
                        ;BUFFERS AND THE PATTERN NUMBER LAST WRITTEN TO THE BUFFER. THE
                        ;NUMBER OF BUFFERS IS IN BUFCNT AND THE SIZE OF THE BUFFERS IS IN
                        ;SIZBUF. THE DATA CAN BE READ INTO THE DM AT ADDRESS IN DATBUF.

001742  104205  002460  READ:   MOV #FREE,R5                  ;GET ADDRESS OF TABLE ENTRIES
001744  104304  002430          MOV BUFCNT,R4                 ;GET COUNT OF BUFFERS

                        ;READ DATA FROM HOST BUFFER INTO DM MEMORY

001746  104303  002433  1$:     MOV     DATBUF,R3             ;R3 = ADDRESS OF DATA BUFFER IN DM MEMORY
001750  104302  002434          MOV     SIZBUF,R2             ;SIZE OF BUFFER
001752  104257                  MOV     (R5)+,R0              ;GET BUFFER ADDRESS
001753  104151                  MOV     (R5),R1
001754  103201  177774          BIC     #177774,R1            ;PLEAR PATTERN NUMBER
001756  104070  002426          MOV     R0,TADR               ;SAVE ADDRESS IN CASE OF ERROR
001760  104010  002427          MOV     R1,TADRH
001762  060013                  XFC     UREAD                 ;READ THE WORD
001763  115001                  TST     R1                    ;WAS IT READ PROPERLY?
001764  012030                  BEQ     3$                    ;IF SO, GO ON TO DO MORE
001765  117401                  DEC     R1                    ;WAS IT AN NXM ERROR?
001766  052010                  BNE     2$                    ;IF NOT, REPORT PARITY ERROR
001767                          ERRHRD  MS1003,TADR,TADRH,SIZBUF
001767  104010  001332                                                  MOV R1,SAVREG
001771  104301  002434                                                  MOV SIZBUF,R1
001773  100461                                                          MOV R1,-(SP)
001774  104301  002427                                                  MOV TADRH,R1
001776  100461                                                          MOV R1,-(SP)
001777  104301  002426                                                  MOV TADR,R1
002001  100461                                                          MOV R1,-(SP)
002002  104301  001332                                                  MOV SAVREG,R1
002004  021250                                                          CALL RERROR     ;ERROR # 1003.
002005  101753                                                          .WORD ERHARD+ERRN
002006  030332                                                          .WORD <PRMS*10000>+MS1003
002007  002030                  BR      3$                    ;EXIT
002010                  2$:     ERRHRD  MS1004,TADR,TADRH,SIZBUF
002010  104010  001332                                                  MOV R1,SAVREG
002012  104301  002434                                                  MOV SIZBUF,R1
002014  100461                                                          MOV R1,-(SP)
002015  104301  002427                                                  MOV TADRH,R1
002017  100461                                                          MOV R1,-(SP)
002020  104301  002426                                                  MOV TADR,R1
002022  100461                                                          MOV R1,-(SP)
002023  104301  001332                                                  MOV SAVREG,R1
002025  021250                                                          CALL RERROR     ;ERROR # 1004.
002026  101754                                                          .WORD ERHARD+ERRN
002027  030452                                                          .WORD <PRMS*10000>+MS1004

                        ; *** SET UP OUT BUFFER IN CASE OF ERROR

002030                  3$:     PUSH    <R5,R4>
002030  100465                                                          MOV R5,-(SP)
002031  100464                                                          MOV R4,-(SP)
002032  104205  002460          MOV #FREE,R5                  ;GET ADDRESS OF TABLE ENTRIES
002034  104304  002430          MOV BUFCNT,R4                 ;GET COUNT OF BUFFERS
002036  106204  000004          CMP     #4,R4                 ;IS R4 > 4?
```

```
                                        BCC     4$                      ;IF NOT, SKIP NEXT INSTRUCTION
                                        MOV     #4,R4                   ;IF SO, FORCE R4 = 4
                                ; *** STORE IN OUTPUT BUFFER JUST IN CASE OF ERROR
                                4$:     MOV     #OUT.05,R0              ;R0 -> OUTPUT BUFFER
                                        MOV     SIZBUF,R1
                                        MOV     R1,(R0)+                ;GIVE BUFFER SIZE
                                        DEC     R4                      ;ADJUST R4
                                        MOV     BTABLE(R4),R1           ;GIVE START OF ALL BUFFERS
                                        MOV     R1,(R0)+
                                        INC     R4                      ;RESTORE R4
                                5$:     MOV     (R5)+,R1                ;STORE BUFFER LO ADDRESS
                                        MOV     R1,(R0)+
                                        MOV     (R5)+,R1                ;STORE BUFFER HI ADDRESS
                                        BIC     #177774,R1
                                        MOV     R1,(R0)+
                                        DEC     R4                      ;DECREMENT (# OF BUFFERS)
                                        BNE     5$                      ;IF NOT DONE, CONTINUE
                                        MOV     R0,OUTPTR               ;OUT POINTER HAS ADDRESS PAST BUFFERS LOCATIONS
                                        ADD     #6,OUTPTR               ;OUTPRT -> PAST DATA OF FIXED LENGTH
                                        MOV     R0,OUTPT2               ;SAVE PREVIOUS POINTER
                                        POP     <R4,R5>
                                                                                        MOV (SP)+,R4
                                                                                        MOV (SP)+,R5

                                ;COMPARE DATA READ WITH DATA PATTERN

                                        MOV     SIZBUF,TEMP1            ;TEMP1 IS DECREMENT COUNTER
                                        MOV     DATBUF,R3               ;R3 -> DM DATA BUFFER
                                        DEC     R5                      ;R5 -> BACK TO LO BUFFER ADDRESS
                                        MOV     (R5)+,TADR              ;GET BUFFER ADDRESS
                                        MOV     (R5),TADRH              ;
                                        BIC     #177774,TADRH           ;MASK HI ADDRESS ONLY
                                        MOV     (R5)+,R2                ;R2 = PATTERN
                                        BIC     #177763,R2              ;MASK PATTERN
                                        ADD     #PAT0,R2                ;R2 -> PATTERN
                                        CLR     ERRNUM                  ;CLEAR ERROR COUNT
                                6$:     MOV     #3,TEMP2                ;TEMP2 = SIZE OF ANY PATTERN
                                7$:     MOV     (R3)+,R0                ;DATA RECEIVED
                                        MOV     (R2)+,R1                ;DATA EXPECTED
                                        CMP     R0,R1                   ;COMPARE THE TWO
                                        BEQ     8$                      ;IF EQUAL, NO ERROR

                                ; *** ERROR HERE

                                        INC     ERRNUM                  ;INCREASE THE ERROR COUNT
                                        CMP     #3,ERRNUM               ;MORE THAN 3 ERROR REPORTS?
                                        BMI     8$                      ;IF SO, DO NOT ADD ANY MORE TO THE ERROR REPORT
                                        PUSH    <R0,R1>
                                                                                        MOV R0,-(SP)
                                                                                        MOV R1,-(SP)
                                        MOV     OUTPTR,R0               ;R0 -> WHERE IN THE OUTPUT BUFFER
                                                                        ; TO PUT ERROR INFO
                                        MOV     TADR,R1
                                        MOV     R1,(R0)+                ;STORE ERROR ADDRESS
                                        MOV     TADRH,R1
                                        MOV     R1,(R0)+                ;
                                        MOV     (SP)+,R1                ;STORE DATA EXPECTED
```

FREE MEMORY CHECK

```
     87 002152  100271                          MOV     R1,(R0)+
     88 002153  104261                          MOV     (SP)+,R1            ;STORE DATA RECEIVED
     89 002154  100271                          MOV     R1,(R0)+
     90 002155  104070  002445                  MOV     R0,OUTPTR           ;STORE NEW OUTPUT POINTER
     91
     92                                ; *** RESUME NORMAL PROCESSING
     93
     94 002157  105200  000002  002426  8$:     ADD     #2,TADR             ;INCREMENT LO ADDRESS
     95 002162  052165                          BNE     9$                  ;IF DID NOT CROSS EXTENDED BOUNDARY, SKIP
     96 002163  115400  002427                  INC     TADRH               ;ELSE, INCREMENT EXTENDED ADDRESS
     97 002165  117400  002443          9$:     DEC     TEMP1               ;GONE OVER THE ENTIRE BUFFER?
     98 002167  012176                          BEQ     10$                 ;IF SO, EXIT LOOP
     99 002170  117400  002444                  DEC     TEMP2               ;AT END OF PATTERN?
    100 002172  052125                          BNE     7$                  ;IF NOT, LOOP BACK
    101 002173  107202  000003                  SUB     #3,R2               ;R2 -> START OF PATTERN
    102 002175  002122                          BR      6$                  ;LOOP BACK TO RESET TEMP2
    103
    104                                ; *** ALL OF THE BUFFER HAS BEEN TESTED
    105
    106 002176  115000  002442          10$:    TST     ERRNUM              ;ANY ERRORS OCCURED?
    107 002200  012251                          BEQ     13$                 ;IF NOT, CHECK IF DONE WITH ROUTINE
    108 002201  104307  002446                  MOV     OUTPT2,R0           ;R0 -> AFTER BUFFER LOCATIONS STORED
    109 002203  104021                          MOV     R2,R1               ;R2 -> PATTERN
    110 002204  107201  002254                  SUB     #PAT0,R1            ;ISOLATE PATTERN #
    111 002206  103201  177763                  BIC     #177763,R1          ;MASK PATTERN NUMBER
    112 002210  105201  000000                  ADD     #0,R1               ;CLEAR CARRY
    113 002212  110601                          ROR     R1
    114 002213  110601                          ROR     R1                  ;PATTERN NUMBER IN LO 2 BITS
    115 002214  100271                          MOV     R1,(R0)+            ;STORE PATTERN NUMBER
    116 002215  104301  002432                  MOV     CURPAT,R1
    117 002217  100271                          MOV     R1,(R0)+            ;LAST PATTERN WRITTEN
    118 002220  104301  002447                  MOV     LBUFWL,R1
    119 002222  100271                          MOV     R1,(R0)+            ;LAST BUFFER WRITTEN
    120 002223  104301  002450                  MOV     LBUFWH,R1
    121 002225  100271                          MOV     R1,(R0)+
    122 002226  104301  002442                  MOV     ERRNUM,R1
    123 002230  100271                          MOV     R1,(R0)+            ;ERROR COUNT
    124 002231  106200  000003  002442          CMP     #3,ERRNUM           ;DO WE HAVE MORE THAN 3 ERRORS?
    125 002234  042240                          BCC     11$                 ;IF NOT, CONTINUE
    126 002235  104202  000002                  MOV     #2,R2               ;SET R2 FOR MAX
    127 002237  002243                          BR      12$                 ;
    128 002240  104302  002442          11$:    MOV     ERRNUM,R2           ;STORE ERROR COUNT IN R2
    129 002242  117402                          DEC     R2                  ;ADJUST R2
    130 002243  104621  002455          12$:    MOV     ETABLE(R2),R1       ;TABLE ENTRY
    131 002245  100171                          MOV     R1,(R0)
    132 002246                                  ERRHRD  MS1005
        002246  021250                                                      CALL RERROR     ;ERROR # 1005.
        002247  101755                                                      .WORD ERHARD+ERRN
        002250  000560                                                      .WORD <PRMS*10000>+MS1005
    133
    134                                ;CYCLE THROUGH ALL HOST BUFFERS
    135
    136 002251  117404          13$:    DEC     R4                  ;COUNT BUFFERS
    137 002252  051746                          BNE     1$                  ;REPEAT FOR ALL BUFFERS
    138 002253  000000          14$:    RETURN
    139
```

```
                                ;DATA PATTERN TABLES

002254  111111          PAT0:   .WORD ^B1001001001001001
002255  044444                  .WORD ^B0100100100100100
002256  022222                  .WORD ^B0010010010010010
002257  000000                  .WORD 0

002260  177400          PAT1:   .WORD ^B1111111100000000
002261  007760                  .WORD ^B0000111111110000
002262  000377                  .WORD ^B0000000011111111
002263  000000                  .WORD 0

002264  155555          PAT2:   .WORD ^B1101101101101101
002265  133333                  .WORD ^B1011011011011011
002266  066666                  .WORD ^B0110110110110110
002267  000000                  .WORD 0

002270  000377          PAT3:   .WORD ^B0000000011111111
002271  170017                  .WORD ^B1111000000001111
002272  177400                  .WORD ^B1111111100000000
```

```
                                        ;CHECK IF TEST ADDRESS IS IN BOUNDS OF READABLE MEMORY

002273  106300  002427  002423  BCHKM:  CMP TADRH,FRADRH            ;COMPARE TEST ADDRESS WITH FIRST READABE ADDRESS
002276  012301                          BEQ 1$                      ;BRANCH IF EQUAL
002277  042306                          BCC 2$                      ;BRANCH IF TEST ADDRESS HIGHER
002300  002415                          BR BCHKMX                   ;BRANCH IF TEST ADDRESS LOWER
002301  106300  002426  002422  1$:     CMP TADR,FRADR
002304  042306                          BCC 2$                      ;BRANCH IF HIGHER OR SAME
002305  002415                          BR BCHKMX                   ;BRANCH IF LOWER
002306  106300  002425  002427  2$:     CMP LRADRH,TADRH            ;COMPARE TEST ADRESS WITH LAST READABLE ADDRESS
002311  012314                          BEQ 3$                      ;BRANCH IF EQUAL
002312  042321                          BCC 4$                      ;BRANCH IF READABLE ADDRESS HIGHER
002313  002415                          BR BCHKMX                   ;BRANCH IF READABLE ADDRESS LOWER
002314  106300  002424  002426  3$:     CMP LRADR,TADR
002317  042321                          BCC 4$                      ;BRANCH IF HIGHER OR SAME
002320  002415                          BR BCHKMX                   ;BRANCH IF LOWER

002321  104305  002416          4$:     MOV FWADR,R5                ;WRITE ONES INTO FIRST ADDRESS
002323  104304  002417                  MOV FWADRH,R4
002325  104207  177777                  MOV #177777,R0
002327  021202                          CALL MWRITE
002330  104305  002426                  MOV TADR,R5                 ;READ FROM TEST ADDRESS
002332  104304  002427                  MOV TADRH,R4
002334  021164                          CALL MREAD
002335  106201  000001                  CMP     #1,R1               ;DID WE GET A NXM?
002337  012415                          BEQ     BCHKMX              ;IF SO, EXIT
002340  106207  177777                  CMP #177777,R0              ;CHECK DATA READ FOR ALL ONES
002342  052415                          BNE BCHKMX                  ;GO TO NEXT BIT IF NOT ALL ONES

002343  104305  002416                  MOV FWADR,R5                ;WRITE ALL ZEROS TO FIRST ADDRESS
002345  104304  002417                  MOV FWADRH,R4
002347  114007                          CLR R0
002350  021202                          CALL MWRITE
002351  104305  002426                  MOV TADR,R5                 ;READ FROM TEST ADDRESS
002353  104304  002427                  MOV TADRH,R4
002355  021164                          CALL MREAD
002356  106201  000001                  CMP     #1,R1               ;DID WE GET A NXM?
002360  012415                          BEQ     BCHKMX              ;IF SO, EXIT
002361  115007                          TST R0                      ;CHECK DATA READ FOR ALL ZEROS
002362  052415                          BNE BCHKMX                  ;GO TO NEXT BIT IF NOT ALL ZEROS

002363  104301  002416                  MOV FWADR,R1                ;COMPUTE XOR OF FIRST ADDRESS
002365  104052                          MOV R5,R2                   ; AND TEST ADDRESS
002366  021223                          CALL XOR
002367  104027                          MOV R2,R0                   ;RESULT TO R0
002370  104301  002417                  MOV FWADRH,R1               ;NOW DO IT FOR HIGH BITS
002372  104042                          MOV R4,R2
002373  021223                          CALL XOR
002374                                  ERRHRD MS1006,FWADR,FWADRH,R5,R4,R0,R2 ;UNIBUS ADDRESSING ERROR. TWO ADDRESSES READ
002374  100462                                                          MOV R2,-(SP)
002375  100467                                                          MOV R0,-(SP)
002376  100464                                                          MOV R4,-(SP)
002377  100465                                                          MOV R5,-(SP)
002400  104010  001332                                                  MOV R1,SAVREG
002402  104301  002417                                                  MOV FWADRH,R1
002404  100461                                                          MOV R1,-(SP)
002405  104301  002416                                                  MOV FWADR,R1
```

```
        002407  100461                          MOV R1,-(SP)
        002410  104301  001332                  MOV SAVREG,R1
        002412  021250                          CALL RERROR     ;ERROR # 1006.
        002413  101756                          .WORD ERHARD+ERRN
        002414  061105                          .WORD <PRMS*10000>+MS1006
     50
     51 002415  000000          BCHKMX: RETURN
```

```
                                ;PROGRAM VARIABLES
                                
002416  000000                  FWADR:  .WORD 0                 ;FIRST ADDRESS CONTAINING ADDRESS DATA
002417  000000                  FWADRH: .WORD 0
002420  000000                  LWADR:  .WORD 0                 ;LAST ADDRESS CONTAINING ADDRESS DATA
002421  000000                  LWADRH: .WORD 0

002422  000000                  FRADR:  .WORD 0                 ;FIRST ADDRESS READABLE
002423  000000                  FRADRH: .WORD 0
002424  000000                  LRADR:  .WORD 0                 ;LAST ADDRESS READABLE
002425  000000                  LRADRH: .WORD 0

002426  000000                  TADR:   .WORD 0                 ;TEST ADDRESS
002427  000000                  TADRH:  .WORD 0

002430  000000                  BUFCNT: .WORD 0                 ;COUNT OF BUFFERS IN HOST MEMORY
002431  000000                  CURBUF: .WORD 0                 ;CURRENT BUFFER BEING WRITTEN IN HOST
002432  000000                  CURPAT: .WORD 0                 ;CURRENT DATA PATTERN BEING WRITTEN
002433  000000                  DATBUF: .WORD 0                 ;ADDRESS OF DATA BUFFER IN DM MEMORY
002434  000000                  SIZBUF: .WORD 0                 ;SIZE OF DATA BUFFER IN HOST MEMORY

                                ;HOST BUFFER TABLE ENTRIES ARE STORED AFTER THE PROGRAM AT FREE.
                                ;       TWO WORDS PER TABLE
                                ;               FIRST WORD  - LOW 16 BITS OF HOST ADDRESS
                                ;               SECOND WORD - BITS 1-0 HIGH TWO BITS OF HOST ADDRESS
                                ;                             BITS 3-2 PATTERN LAST WRITTEN

002435  000000                  CMPSIZ: .WORD 0                 ;TABLE FOR COMPARE DATA PATTERN XFC
002436  000000                  CMPADR: .WORD 0
002437  000000                  CMP1:   .WORD 0
002440  000000                  CMP2:   .WORD 0
002441  000000                  CMP3:   .WORD 0

                                ;       MISSELENEOUS DATA STORAGE
002442  000000                  ERRNUM: .WORD 0                 ;HOLDS NUMBER OF ERRORS DURING COMPARE
002443  000000                  TEMP1:  .WORD 0                 ;TEMPORARY STORAGE
002444  000000                  TEMP2:  .WORD 0                 ; SAME
002445  000000                  OUTPTR: .WORD 0                 ;POINTER INTO OUTPUT BUFFER FOR COMPARE
002446  000000                  OUTPT2: .WORD 0                 ; SAME
002447  000000                  LBUFWL: .WORD 0                 ;LAST BUFFER WRITTEN SAVED IN WRITE
002450  000000                  LBUFWH: .WORD 0                 ; SAME
002451  001274  001264  001254  BTABLE: .WORD B1,B2,B3,B4
002454  001244
002455  001357  001332  001305  ETABLE: .WORD E1,E2,E3
```

```
                                ;MESSAGE STORAGE OVERLAY
                                
002460                          FREE:                                   ; FREE SPACE BEGINS HERE
002460                                  DMOVLY MS,0
002460  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                        .NLIST BEX
                                
000000  042  116  117  MS1000: .ASCII\'NON-EXISTANT MEMORY ERROR TRYING TO READ FROM UNIBUS.'N\
000034  123  062  065          .ASCII\S25'OCTAL'S6'HEX'N\
000045  123  070  042          .ASCII\S8'ADDRESS'S9018R9R9S5H18N\
000062  000                    .BYTE 0
000063  042  120  101  MS1001: .ASCII\'PARITY ERROR ON READ FROM UNIBUS.'N\
000105  123  062  065          .ASCII\S25'OCTAL'S6'HEX'N\
000116  123  070  042          .ASCII\S8'ADDRESS'S9018R9R9S5H18N\
000133  123  070  042          .ASCII\S8'DATA READ'S7016R9S5H16N\
000150  123  070  042          .ASCII\S8'DATA EXPECTED'S3016R9S5H16N\
000167  000                    .BYTE 0
000170  042  125  116  MS1002: .ASCII\'UNIBUS ADDRESSING ERROR - INCORRECT DATA READ.'N\
000220  042  115  105          .ASCII\'MEMORY LOCATION SHOULD CONTAIN OWN ADDRESS.'N\
000247  123  062  065          .ASCII\S25'OCTAL'S6'HEX'N\
000260  123  070  042          .ASCII\S8'ADDRESS'S9018R9R9S5H18N\
000275  123  070  042          .ASCII\S8'DATA READ'S7016R9S5H16N\
000312  123  070  042          .ASCII\S8'DATA EXPECTED'S3016R9S5H16N\
000331  000                    .BYTE 0
000332  042  116  117  MS1003: .ASCII\'NON-EXISTANT MEMORY ERROR TRYING TO READ FROM UNIBUS WITHIN BUFFER.'N\
000375  123  062  070          .ASCII\S28'OCTAL'S6'HEX'N\
000406  042  040  123          .ASCII\" STARTING ADDESS OF BUFFER '018R9R9S5H18N\
000433  123  070  042          .ASCII\S8'BUFFER SIZE'S8016R9S5H16N\
000451  000                    .BYTE 0
000452  042  120  101  MS1004: .ASCII\'PARITY ERROR ON READ FROM UNIBUS WITHIN BUFFER.'N\
000503  123  062  070          .ASCII\S28'OCTAL'S6'HEX'N\
000514  042  040  123          .ASCII\" STARTING ADDESS OF BUFFER '018R9R9S5H18N\
000541  123  070  042          .ASCII\S8'BUFFER SIZE'S8016R9S5H16N\
000557  000                    .BYTE 0
000560  042  104  101  MS1005: .ASCII\'DATA COMPARE FAILED AFTER WRITE THEN READ FROM UNIBUS.'N\
000614  042  040  1[illegible] .ASCII\" BUFFER SIZE = '016''(O)'R9S3H16''(X)'R9S3D16''.(D)'N\
000646  042  040  123          .ASCII\" STARTING ADDRESSES OF BUFFERS'N\
000666  123  066  042          .ASCII\S6'OCTAL'S11'HEX'NR1\
000700  042  040  103          .ASCII\" CURRENT DATA PATTERN READ'S17D6N\
000721  042  040  114          .ASCII\" LAST PATTERN WRITTEN'S22D6N\
000740  042  040  123          .ASCII\" STARTING ADDRESS OF LAST BUFFER WRITTEN'S3018''(O)'R9R9S3H18''(X)'N\
001001  042  040  116          .ASCII\" NUMBER OF ERRORS FOUND'S20D16''.(D)'N\
001024  123  064  042          .ASCII\S4'LOCATION'S6'DATA EXPECTED'S6'DATA RECEIVED'N\
001054  123  062  042          .ASCII\S2'OCTAL   HEX'S6'OCTAL   HEX'S8'OCTAL   HEX'N\
001103  122  061                .ASCII\R1\
001104  000                    .BYTE 0
001105  042  125  116  MS1006: .ASCII\'UNIBUS ADDRESSING ERROR. TWO ADDRESSES READ SAME LOCATION.'N\
001143  123  063  063          .ASCII\S33'OCTAL'S6'HEX'N\
001154  123  070  042          .ASCII\S8'KNOWN GOOD ADDRESS'S6018R9R9S5H18N\
001177  123  070  042          .ASCII\S8'ERROR ADDRESS'S11018R9R9S5H18N\
001217  123  070  042          .ASCII\S8'ADDRESS BIT IN ERROR'S4018R9R9S5H18N\
001243  000                    .BYTE 0
                                
001244  123  065  117  B4:     .ASCII\S5018R9R9S10H18N\
001254  123  065  117  B3:     .ASCII\S5018R9R9S10H18N\
001264  123  065  117  B2:     .ASCII\S5018R9R9S10H18N\
001274  123  065  117  B1:     .ASCII\S5018R9R9S10H18N\
```

```
     57 001304      000                              .BYTE 0
     58
     59 001305      123     061     117  E3:         .ASCII\S1018S2R9R9H18S4016R9S2H16S7016R9S2H16S11N\
     60 001332      123     061     117  E2:         .ASCII\S1018S2R9R9H18S4016R9S2H16S7016R9S2H16S11N\
     61 001357      123     061     117  E1:         .ASCII\S1018S2R9R9H18S4016R9S2H16S7016R9S2H16S11N\
     62 001404      000                              .BYTE 0
     63
     64 001404                                       DMEND
        001405  000105                               .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
     65         000001                               .END
```

SYMBOL TABLE

```
ACHK1   001415          DRVID = 000004          HDRPRE= 000005          LARGE = 000001          OUT.21  001033
ACHK2   001440          DRVONL= 000213          HD.BAD= 110000          LBHINB= 177417          OUT.22  001034
ACHK3   001451          DRVRUN= 000014          HD.DBN= 140000          LBLONB= 177760          OUT.23  001C35
ACHK4   001457          DU.DFL= 020000          HD.LBN= 000000          LBNCYL= 000000          OUT.24  001036
ARGS$ = 000007          DU.FTL= 050000          HD.PRV= 050000          LBNHST= 000012          OUT.25  001037
ATTN  = 000002          DU.INF= 030000          HD.RBN= 060000          LBNTRK= 000011          OUT.26  001040
AVAIL = 000100          DU.QUE= 010000          HD.REV= 030000          LBUFWH  002450          OUT.27  001041
BCHK    001464          DU.SPC= 060000          HD.XBN= 120000          LBUFWL  002447          OUT.28  001042
BCHKM   002273          DU.TER= 040000          HEADER= 000002          LINKLN= 000007          OUT.29  001043
BCHKMX  002415          D.LIMT= 000001          HIBYTE= 177400          LOBYTE= 000377          OUT.30  001044
BCHK1   001472          D.SCHR= 000002          HICYL = 000001          LONG    001374          OUT.31  001045
BCHK2   001511          ECC   = 000015          HIDBN = 000003          LONGTO= 000001          OUT.32  001046
BEUSED= 000040          ECCCHK= 010000          HILBN = 000002          LOW   = 000002          OUT.33  001047
BF.DAT= 000000          ECCFLG= 010000          HIMEM = 010000          LRADR   002424          OUT.34  001050
BF.ECC= 000401          ECCRSH= 000002          HIRBN = 000003          LRADRH  002425          OVERFL= 000002
BF.EDC= 000400          ECHO  = 000010          HIXBN = 000002          LUNIT   001331          OVE.MN= 000714
BIT00 = 000001          ECHOC = 000350          HOSTRQ  000751          LWADR   002420          OVE.MS= 000000
BIT01 = 000002          EOC   = 100000          HRDREV= 000003          LWADRH  002421          OVL.MN= 001545
BREAK = 000000          ERECOV= 000006          INITW = 040000          MEDTYP= 000006          OVL.MS= 001406
BTABLE  002451          ERHARD= 100000          INSEEK= 000012          MESSAG= 060015          OVL...= 003153
BUFCNT  002430          ERLEV = 000002          INTEDC= 000105          MICREV= 000003          OVSTRT= 007774
BUFFLG= 040000          ERRMC = 060014          IN.RQ   001051          MRD   = 000016          OVS.MN= 001040
BUFSIZ= 000043          ERRMES= 060013          IN.01   001052          MREAD   001164          OVS.MS= 004352
B1      001274          ERRN  = 001757          IN.02   001053          MS1000  000000          OV... = 003573
B2      001264          ERRNUM  002442          IN.03   001054          MS1001  000063          PAT0    002254
B3      001254          ERSOFT= 140000          IN.04   001055          MS1002  000170          PAT1    002260
B4      001244          ETABLE  002455          IN.05   001056          MS1003  000332          PAT2    002264
CCHK    001524          EXIT  = 000021          IN.06   001057          MS1004  000452          PAT3    002270
CHECK = 000010          E1      001357          IN.07   001060          MS1005  000560          PHYSA = 001000
CHGMOD= 000201          E2      001332          IN.08   001061          MS1006  001105          PRMS  = 000006
CHRRES= 000170          E3      001305          IN.09   001062          MWR   = 000017          PTYPE$= 000030
CLRBUF  000777          FB.DAT= 000000          IN.10   001063          MWRITE  001202          RBNBN = 000200
CMPADR  002436          FB.EDC= 000400          IN.11   001064          NEWSUB= 004000          RBNTRK= 000004
CMPSIZ  002435          FCTSIZ= 000010          IN.12   001065          ONLYCL= 000200          RBUFLN= 000415
CMP1    002437          FIRSTU= 007717          IN.13   001066          OUTPTR  002445          RCLB  = 000001
CMP2    002440          FMEMFL  001405          IN.14   001067          OUTPT2  002446          RCONT = 000000
CMP3    002441          FORMAT= 000001          IN.15   001070          OUT.RQ  001006          RCTCPS= 000001
COMPAR= 000006          FRADR   002422          IN.16   001071          OUT.01  001007          RCTCSZ= 000014
COMPLT= 000176          FRADRH  002423          IN.17   001072          OUT.02  001010          RCV   = 000005
CURBUF  002431          FRAME = 000004          IN.18   001073          OUT.03  001011          RCVERR= 000004
CURPAT  002432          FREE    002460          IN.19   001074          OUT.04  001012          RCVRDY= 000001
CVT   = 000020          FSTOP = 100000          IN.20   001075          OUT.05  001013          RDSTAT  000720
DATBUF  002433          FTIME = 001000          IN.21   001076          OUT.06  001014          READ    001742
DATCMP= 000002          FTLDEV= 040000          IN.22   001077          OUT.07  001015          REDWRT= 000100
DATERR= 000020          FTLSYS= 000000          IN.23   001100          OUT.08  001016          REGS$ = 177777
DATPRE= 000005          FT.BUF= 000000          IN.24   001101          OUT.09  001017          REGU$ = 000001
DBNCYL= 000022          FT.HI = 000001          IN.25   001102          OUT.10  001020          RERRCA  001315
DCMPAL= 000001          FT.LOW= 000002          IN.26   001103          OUT.11  001021          RERROR  001250
DCYLS = 020000          FWADR   002416          IN.27   001104          OUT.12  001022          RERRPA  001311
DINIT = 000011          FWADRH  002417          IN.28   001105          OUT.13  001023          RESEEK= 020000
DIREC = 010000          GETCHR= 000207          IN.29   001106          OUT.14  001024          RETRY = 000004
DISCON= 000204          GETSTA= 000011          IN.30   001107          OUT.15  001025          RETS  = 00000?
DONE  = 060016          GETSUB= 000210          IN.31   001110          OUT.16  001026          REVEC = 00040?
DONECD  001662          GRPCYL= 000002          IN.32   001111          OUT.17  001027          REVECT= 000004
DROP  = 100000          GRPOFF= 000011          IN.33   001112          OUT.18  001030          REVINP= 000040
DRTYPE= 000007          HBHINB= 007777          IN.34   001113          OUT.19  001031          REVS  = 000007
DRVCLR= 000005          HBLONB= 170377          IRECLB= 000216          OUT.20  001032          RM    = 000004
```

SYMBOL TABLE

```
RONLY = 004000          STSRES= 000366          S.SDCL= 000001          UDA50 = 000001          U.RBN = 000041
RREAL = 013400          ST.C  = 000002          S.TGOF= 000013          UDA52 = 000000          U.RTRY= 000025
RSTOP = 100000          ST.CON= 000002          S.TGSS= 000015          UNIT0 = 000001          U.RVER= 000046
RTRIES= 001000          ST.DB = 001000          S.TRKL= 000004          UN'T1 = 000002          U.RWER= 000045
RWRDY = 100000          ST.DF = 000020          TADR    002426          UNIT2 = 000004          U.RWTO= 000006
RW.ANG= 000006          ST.DR = 000040          TADRH   002427          UNIT3 = 000010          U.SDIL= 000031
RW.BUF= 000001          ST.ERR= 000002          TALKER  001114          UNSSUC= 000175          U.SDIS= 000030
RW.CMD= 000004          ST.FE = 000200          TALK1A  001126          UREAD = 000013          U.SEEK= 000007
RW.HI = 000003          ST.FO = 002000          TALK1B  001136          UTOTST= 060012          U.SNUM= 000047
RW.LOW= 000002          ST.MOD= 000001          TALK2A  001156          UWRITE= 000014          U.SUBP= 000001
RW.SDI= 000005          ST.MSK= 000000          TALK2B  001157          U.CBN = 000037          U.SUBU= 000034
RW.STA= 000000          ST.OA = 000200          TEMP1   002443          U.CCNT= 000012          U.TIMO= 000005
SAVREG  001332          ST.PE = 000040          TEMP2   002444          U.CCOP= 000044          U.TSEC= 000020
SBCRES= 000167          ST.PS = 000002          TEST4 = 000000          U.CCYL= 000053          U.WPRT= 000032
SCTWRD= 000377          ST.RE = 000100          TIMEOU= 000001          U.CGRP= 000055          U.WRIT= 000023
SDILTO  001163          ST.RR = 000100          TLEN.U= 000061          U.COPY= 000043          WAITSI= 000012
SDISTO  001162          ST.RTY= 000003          TO      001233          U.CSEC= 000021          WBUFLN= 000401
SDIVER= 000000          ST.RU = 000001          TOOBIG= 000001          U.CTRK= 000015          WCHECK= 000010
SEKINP= 002000          ST.SR = 000020          TRACKS= 000020          U.ECCT= 000027          WCHKAL= 000004
SEND  = 000004          ST.STA= 000001          TRKGRP= 000003          U.ELEV= 000024          WCONT = 040000
SEQSEK= 000100          ST.S7 = 000400          T1MSIZ= 060000          U.LCYL= 000056          WONLY = 002000
SHRTTO= 000000          ST.UNT= 000000          T2CMD = 060002          U.LGRP= 000060          WREAL = 122400
SIZBUF  002434          ST.WE = 000010          T2DLL = 060001          U.MASK= 000022          WRITE   001666
SNDAGN  000756          SUB   = 000013          T4BB1 = 060005          U.MBN = 000035          WRONG = 000002
SPADJU  001326          S.BADP= 000010          T4BB2 = 060006          U.MLEV= 000026          WSTOP = 140000
SS    = 000001          S.BESS= 000011          T4MPRM= 060003          U.MSEC= 000017          XBNCYL= 000021
STACK   001373          S.MCNT= 000011          T4MXFR= 060011          U.NEXT= 000000          XFERRT= 000000
START   001374          S.MEGR= 000006          T4SEEK= 060010          U.NFUN= 000010          XMTERR= 000400
STATEX  000746          S.MEGW= 000007          T4SOFT= 060007          U.NSEC= 000016          XOR     001223
STATLP  000724          S.PARM= 000000          T4UPRM= 060004          U.PARM= 000033          XREAD = 000002
STATOK  000737          S.PAT = 000003          UDADM1= 001000 G        U.PAT = 000011          XWRITE= 000003
STATUS= 000007          S.SCHR= 000005          UDATA   001222          U.PCTG= 000014

. ABS.  007366      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  4487 WORDS  ( 18 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  71 PAGES
,B:UDAT1.L50/C=$DMACRO,B:UDAT1
```

CROSS REFERENCE TABLE (CREF V04.00)

```
DROP      8-10#
DRTYPE    5-15#
DRVCLR    6-35#
DRVID     5-14#
DRVONL    6-33#
DRVRUN    6-34#
DU.DFL    3-50#
DU.FTL    3-53#
DU.INF    3-51#
DU.QUE    3-49#
DU.SPC    3-54#     6-3       6-4       6-5       6-6       6-7       6-8       6-9       6-10      6-11      6-12      6-13      6-14      6-15
          6-16      6-17
DU.TER    3-52#
E1       39-43     40-61#
E2       39-43     40-60#
E3       39-43     40-59#
ECC       4-16#
ECCCHK    8-28#
ECCFLG    3-88#
ECCRSH    5-11#
ECHO      4-11#
ECHOC     6-46#
EOC       3-89#
ERECOV    6-31#
ERHARD    6-52#     30-27     30-29     30-36     36-24     36-26     36-132    38-49
ERLEV     5-10#
ERRMC     6-15#
ERRMES    6-14#     28-39
ERRN     30-3#      30-27     30-27     30-27     30-27#    30-29     30-29     30-29     30-29#    30-36     30-36     30-36     30-36#    36-24
          36-24     36-24     36-24#    36-26     36-26     36-26     36-26#    36-132    36-132    36-132    36-132#   38-49     38-49     38-49
          38-49#
ERRNUM   36-67*     36-76*    36-77*    36-106*   36-122    36-124*   36-128    39-35#
ERSOFT    6-53#
ETABLE   36-130     39-43#
EXIT      4-20#
FB.DAT    3-65#
FB.EDC    3-66#
FCTSIZ    5-36#
FIRSTU    7-11#
FMEMFL   30-13#     30-16
FORMAT    4-4#
FRADR    38-7*      39-8#
FRADRH   38-3*      39-9#
FRAME     5-66#
FREE     32-18      32-40     35-34     36-7      36-31     40-3#
FSTOP     3-85#
FT.BUF    3-58#
FT.HI     3-59#
FT.LOW    3-60#
FTIME     8-16#
FTLDEV    6-51#
FTLSYS    6-50#
FWADR    30-11      30-20     31-8      31-15     32-23     32-41     38-18     38-30     38-42     38-49     38-49     39-3#
FWADRH   30-21      31-4      31-19     32-26     32-42     38-19     38-31     38-46     38-49     38-49     39-4#
GETCHR    6-36#
GETSTA    6-38#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
IN.34   22-77#
IN.RQ   21-19    22-43#   22-78
INITW    8-11#
INSEEK   6-40#
INTEDC   7-9#
IRECLB   6-39#
LARGE    5-74#
LBHINB   5-58#
LBLONB   5-59#
LBNCYL   5-23#
LBNHST   5-39#
LBNTRK   5-37#
LBUFWH  35-39*   36-120   39-41#
LBUFWL  35-38*   36-118   39-40#
LINKLN   3-113#
LOBYTE   5-55#
LONG    23-20    30-7#
LONGTO   5-6#
LOW      5-70#
LRADR   38-14    39-10#
LRADRH  38-10    39-11#
LUNIT   28-19    28-46#
LWADR   30-40    32-21    32-53    39-5#
LWADRH  30-42    32-22    32-50    39-6#
MEDTYP   5-35#
MESSAG   6-16#
MICREV   5-12#
MRD      4-17#   21-16
MREAD   24-10#   30-22    38-24    38-36
MS1000  30-27    40-7#
MS1001  30-29    40-11#
MS1002  30-36    40-17#
MS1003  36-24    40-24#
MS1004  36-26    40-29#
MS1005  36-132   40-34#
MS1006  38-49    40-46#
MWR      4-18#   21-21
MWRITE  25-10#   38-21    38-33
NEWSUB   8-14#
ONLYCL   8-32#
OUT.01  21-22    21-23    22-6#    28-13
OUT.02  22-7#
OUT.03  22-8#
OUT.04  22-9#
OUT.05  22-10#   36-37
OUT.06  22-11#
OUT.07  22-12#
OUT.08  22-13#
OUT.09  22-14#
OUT.10  22-15#
OUT.11  22-16#
OUT.12  22-17#
OUT.13  22-18#
OUT.14  22-19#
OUT.15  22-20#
OUT.16  22-21#
```

```
OUT.17  22-22#
OUT.18  22-23#
OUT.19  22-24#
OUT.20  22-25#
OUT.21  22-26#
OUT.22  22-27#
OUT.23  22-28#
OUT.24  22-29#
OUT.25  22-30#
OUT.26  22-31#
OUT.27  22-32#
OUT.28  22-33#
OUT.29  21-23    22-34#
OUT.30  22-35#
OUT.31  22-36#
OUT.32  22-37#
OUT.33  22-38#
OUT.34  22-39#
OUT.RQ  21-13*   21-14    22-5#
OUTPT2  36-53*   36-108   39-39#
OUTPTR  36-51*   36-52*   36-80    36-90*   39-38#
OV...    3-17     3-17     3-17#   40-4     40-4     40-4     40-4#    40-64    40-64#
OVE.MN   3-17#   40-4
OVE.MS  40-4#    40-64
OVERFL   5-75#
OVL...   3-17     3-17#   40-4     40-4#    40-64    40-64#
OVL.MN   3-17     3-17    40-4     40-4     40-4#
OVL.MS  40-64    40-64    40-64#
OVS.MN   3-17#
OVS.MS  40-4#
OVSTRT   3-42#
PAT0    35-13    36-66    36-110   37-3#
PAT1    37-8#
PAT2    37-13#
PAT3    37-18#
PHYSA    3-17     3-17     3-17     3-17     3-17#
PRMS    30-27    30-27    30-27    30-27    30-27    30-27    30-27    30-27    30-27    30-27    30-27#   30-29    30-29    30-29
        30-29    30-29    30-29    30-29    30-29    30-29    30-29    30-29#   30-36    30-36    30-36    30-36    30-36    30-36
        30-36    30-36    30-36    30-36    30-36#   36-24    36-24    36-24    36-24    36-24    36-24    36-24    36-24    36-24
        36-24    36-24#   36-26    36-26    36-26    36-26    36-26    36-26    36-26    36-26    36-26    36-26    36-26#   36-132
        36-132   36-132   36-132   36-132   36-132   36-132   36-132   36-132   36-132   36-132#  38-49    38-49    38-49    38-49
        38-49    38-49    38-49    38-49    38-49    38-49    38-49#
PTYPE$  30-27    30-27    30-27    30-27    30-27#   30-27#   30-29    30-29    30-29    30-29    30-29    30-29    30-29    30-29
        30-29#   30-29#   30-29#   30-29#   30-36    30-36    30-36    30-36    30-36    30-36    30-36    30-36    30-36#   30-36#
        30-36#   30-36#   36-24    36-24    36-24    36-24    36-24    36-24    36-24#   36-24#   36-24#   36-26    36-26    36-26
        36-26    36-26    36-26    36-26#   36-26#   36-26#   38-49    38-49    38-49    38-49    38-49    38-49    38-49    38-49
        38-49    38-49    38-49    38-49    38-49#   38-49#   38-49#   38-49#   38-49#   38-49#
RBNBN    8-18#
RBNTRK   5-31#
RBUFLN   3-112#
RCLB     8-23#
RCONT    3-84#
RCTCPS   5-8#
RCTCSZ   5-40#
RCV      4-8#    23-27
RCVERR   6-23#   20-47
```

CROSS REFERENCE TABLE (CREF V04.00)

```
RCVRDY   6-21#
RDSTAT  20-32#
READ    33-11    33-22    36-7#
REDWRT   8-19#
REGS$   30-27    30-27    30-27    30-27#   30-29    30-29    30-29    30-29    30-29    30-29#   30-36    30-36    30-36    30-36
        30-36    30-36#   36-24    36-24    36-24    36-24    36-24    36-24    36-24    36-24    36-24    36-24#   36-24#   36-24#
        36-26    36-26    36-26    36-26    36-26    36-26    36-26    36-26    36-26    36-26#   36-26#   36-26#   36-132   36-132#
        38-49    38-49    38-49    38-49    38-49    38-49    38-49    38-49    38-49    38-49    38-49    38-49#   38-49#   38-49#
REGU$   36-24    36-24    36-24    36-24    36-24#   36-24#   36-24#   36-26    36-26    36-26    36-26    36-26#   36-26#   36-26#
        38-49    38-49    38-49    38-49#   38-49#
RERRCA  28-33    28-38#
RERROR  28-6#    30-27    30-29    30-36    36-24    36-26    36-132   38-49
RERRPA  28-34#   28-37
RESEEK   8-12#
RETRY    8-22#
RETS     5-7#
REVEC    8-17#
REVECT   5-63#
REVINP   8-20#
REVS     5-16#
RM       5-32#
RONLY    8-29#
RREAL    3-87#
RSTOP    3-83#
RTRIES   8-31#
RW.ANG   3-77#
RW.BUF   3-72#
RW.CMD   3-75#
RW.HI    3-74#
RW.LOW   3-73#
RW.SDI   3-76#
RW.STA   3-71#
RWRDY    6-26#
S.BADP   7-62#    7-63
S.BESS   7-63#    7-68
S.MCNT   7-68#    7-69
S.MEGR   7-60#    7-61
S.MEGW   7-61#    7-62
S.PARM   7-55#    7-56
S.PAT    7-57#    7-58
S.SCHR   7-59#    7-60
S.SDCL   7-56#    7-57
S.TGOF   7-69#    7-70
S.TGSS   7-70#
S.TRKL   7-58#    7-59
SAVREG  28-48#   36-24    36-24*   36-26    36-26*   38-49    38-49*
SBCRES   6-44#
SCTWRD   7-8#
SDILTO  23-24    23-43#
SDISTO  23-22    23-42#
SDIVER   5-4#
SEKINP   8-15#
SEND     4-7#    23-15
SEQSEK   8-34#
SHRTTO   5-3#
SIZBUF  32-62*   35-18    35-42    36-13    36-24    36-24    36-26    36-26    36-38    36-58    39-20#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
SNDAGN  21-14#   21-18
SPADJU  28-32*   28-42#
SS       5-9#
ST.C     4-30#
ST.CON   4-29#
ST.DB    4-47#
ST.DF    4-44#
ST.DR    4-37#
ST.ERR   4-28#
ST.FE    4-41#
ST.FO    4-46#
ST.MOD   4-27#
ST.MSK   4-25#
ST.OA    4-35#
ST.PE    4-43#
ST.PS    4-39#
ST.RE    4-42#
ST.RR    4-36#
ST.RTY   4-31#
ST.RU    4-40#
ST.S7    4-48#
ST.SR    4-38#
ST.STA   4-26#
ST.UNT   4-24#
ST.WE    4-45#
STACK   20-19    29-5#
START   20-30    30-8#
STATEX  20-46    20-48    20-52#
STATLP  20-39#   20-44    20-50
STATOK  20-42    20-47#
STATUS   4-10#   20-39
STSRES   6-45#
SUB      5-22#
T1MSIZ   6-3#    30-8
T2CMD    6-5#
T2DLL    6-4#
T4BB1    6-8#
T4BB2    6-9#
T4MPRM   6-6#
T4MXFR   6-12#
T4SEEK   6-11#
T4SOFT   6-10#
T4UPRM   6-7#
TADR    31-10*   31-16*   36-17*   36-24    36-24    36-26    36-26    36-61*   36-82    36-94*   38-7     38-14*   38-22    38-34
        39-13#
TADRH   31-5*    31-21*   36-18*   36-24    36-24    36-26    36-26    36-62*   36-63*   36-84    36-96*   38-3     38-10*   38-23
        38-35    39-14#
TALK1A  23-17    23-20#
TALK1B  23-23    23-25#   23-35
TALK2A  23-29    23-38#
TALK2B  23-19    23-33    23-37    23-39#
TALKER  23-12#
TEMP1   36-58*   36-97*   39-36#
TEMP2   36-68*   36-99*   39-37#
TEST4    3-1#
TIMEOU   5-61#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
TLEN.U    7-10#    7-11
TO       27-1#
TOOBIG    5-69#
TRACKS    8-36#
TRKGRP    5-28#
U.CBN     7-41#    7-42
U.CCNT    7-22#    7-23
U.CCOP    7-44#    7-45
U.CCYL    7-48#    7-49
U.CGRP    7-49#    7-50
U.COPY    7-43#    7-44
U.CSEC    7-28#    7-29
U.CTRK    7-24#    7-25
U.ECCT    7-34#    7-35
U.ELEV    7-31#    7-32
U.LCYL    7-50#    7-51
U.LGRP    7-10     7-51#
U.MASK    7-29#    7-30
U.MBN     7-40#    7-41
U.MLEV    7-33#    7-34
U.MSEC    7-26#    7-27
U.NEXT    7-15#    7-16
U.NFUN    7-20#    7-21
U.NSEC    7-25#    7-26
U.PARM    7-38#    7-39
U.PAT     7-21#    7-22
U.PCTG    7-23#    7-24
U.RBN     7-42#    7-43
U.RTRY    7-32#    7-33
U.RVER    7-46#    7-47
U.RWER    7-45#    7-46
U.RWTO    7-18#    7-19
U.SDIL    7-36#    7-37
U.SDIS    7-35#    7-36
U.SEEK    7-19#    7-20
U.SNUM    7-47#    7-48
U.SUBP    7-16#    7-17
U.SUBU    7-39#    7-40
U.TIMO    7-17#    7-18
U.TSEC    7-27#    7-28
U.WPRT    7-37#    7-38
U.WRIT    7-30#    7-31
UDA50     2-31#    2-46     3-14     3-39     20-10
UDA52     2-32#    2-37     3-3      3-34     20-5     20-21     39-46
UDADM1    3-17#
UDATA    24-14    24-16    25-11*   25-15    25-20#
UNIT0     5-46#
UNIT1     5-47#
UNIT2     5-48#
UNIT3     5-49#
UNSSUC    6-42#
UREAD     4-14#    24-15    36-19
UTOTST    6-13#
UWRITE    4-15#    25-16    35-44
WAITSI    4-13#
WBUFLN    3-111#   3-112
```

CROSS REFERENCE TABLE (CREF V04.00)

```
WCHECK    8-37#
WCHKAL    8-39#
WCONT     3-82#
WONLY     8-30#
WREAL     3-86#
WRITE    33-6      33-21     35-9#
WRONG     5-65#
WSTOP     3-81#
XBNCYL    5-41#
XFERRT    5-5#
XMTERR    6-25#    20-41
XOR      26-10#    31-9      31-20     38-44     38-48
XREAD     4-5#
XWRITE    4-6#
```

UDAT2 DISK RESIDENT DMACR X04.01 19-AUG-82 13:07:09 SEQ 0132

TABLE OF CONTENTS

```
                                .TITLE  UDAT2   DISK RESIDENT
                        ;
                        ;
                        ;COPYRIGHT (C)  1981
                        ;DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ;THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ;SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ;OF   THE   ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ;COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ;TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ;WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ;THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ;THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ;NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ;EQUIPMENT CORPORATION.
                        ;
                        ;DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ;ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
000000                  UDA50=0
000001                  UDA52=1
                        ;THIS PROGRAM SHALL BE ASSEMBLED WITH THE PROGRAM DMACRO
                        ;USING A COMMAND LINE SIMILAR TO:
                        ;
                        ;  UDAT2,UDAT2/C=[30,30]DMAC52,UDAT2
                        ;
                        ;LINK USING A COMMAND LINE SIMILAR TO:
                        ;
                        ;  UDAT2.BIN/L=UDAT2
                        ;
                              .MCALL  DMCODE,DMEND,DMOVLY                       ;UDA52
                              .MCALL  JMP,BR,BEQ,CALL,BPL,BCC,BNE,BMI,RETURN    ;UDA52
                              .MCALL  DMODT                                     ;UDA52
                        ;
000000                        DMCODE  UDADM2,0,1364,3,0,1000
                              .SBTTL  UDA DM PROGRAM PARAMETERS
                        ;
                              .LIST MEB
003720                        ERRN=2000.                          ;START ERROR NUMBERS AT 2000.
                        ;
                        ;     EQUATES
                        ;
                        ;     HIGHEST USABLE LOCATION OF UDA MEMORY + 1
                        ;
037777                  HIMEM   =       37777
```

```
                                ;
                                ;
                                ;       OFFSETS FOR FORMAT TRACK TABLE
                                ;
000000                          FT.BUF  =       0.                      ;BUFFER POINTER OFFSET
000001                          FT.HI   =       1.                      ;HI ORDER HEADER OFFSET
000002                          FT.LOW  =       2.                      ;LOW ORDER HEADER OFFSET
                                ;
                                ;
                                ;       OFFSETS FOR FORMAT TRACK BUFFER
                                ;
000000                          FB.DAT  =         0.                    ;FIRST DATA WORD OFFSET
000400                          FB.EDC  =       256.                    ;EDC WORD OFFSET
                                ;
                                ;
                                ;       OFFSETS FOR READ/WRITE I/O CHAIN TABLES
                                ;
000000                          RW.STAT =       0.                      ;STATUS AND NEXT BUFFER POINTER OFFSET
000001                          RW.BUF  =       1.                      ;POINTER TO DATA BUFFER
000002                          RW.LOW  =       2.                      ;HI ORDER EXPECTED HEADER
000003                          RW.HI   =       3.                      ;LOW ORDER EXPECTED HEADER
000004                          RW.CMD  =       4.                      ;SDI COMMAND AND HEAD ADDRESS
000005                          RW.SDI  =       5.                      ;DUMMY SDI CONTROL BLOCK POINTER
000006                          RW.ANG  =       6.                      ;THETA FROM INDEX
                                ;
                                ;       CONSTANTS FOR READ AND WRITE XFC'S
                                ;
140000                          WSTOP   =       140000                  ; LAST ENTRY IN CHAIN FOR WRITE
040000                          WCONT   =       40000                   ; WRITE CONTINUE
100000                          RSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR READ
000000                          RCONT   =       0                       ; READ CONTINUE
100000                          FSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR FORMAT
122400                          WREAL   =       122400                  ; WRITE REAL TIME ECOMMAND
013400                          RREAL   =       13400                   ; READ REAL TIME COMMAND
010000                          ECCFLG  =       10000                   ; ECC ERROR IN BUFFER BIT
100000                          EOC     =       100000                  ; END OF CHAIN
040000                          BUFFLG  =       40000                   ; BUFFER FULL OR EMPTY FLAG
                                ;
                                ;
                                ;       HEADER CODES
                                ;
000000                          HD.LBN  =       000000                  ;GOOD LBN
060000                          HD.RBN  =       060000                  ;GOOD RBN, PERHAPS UNUSED
030000                          HD.REV  =       030000                  ;REVECTORED LBN
110000                          HD.BAD  =       110000                  ;BAD BLOCK
050000                          HD.PRV  =       050000                  ;PRIMARY REVECTORED BLOCK
120000                          HD.XBN  =       120000                  ;XBN BLOCK
140000                          HD.DBN  =       140000                  ;DBN BLOCK
                                ;
                                ;       OFFSETS FOR DATA BUFFERS
                                ;
000000                          BF.DAT  =         0.                    ;DATA
000400                          BF.EDC  =       256.                    ;ERROR DETECTION CODE
000401                          BF.ECC  =       257.                    ;LAST 17 ECC RESIDUES
                                ;
                                ;       BUFFER AND READ/WRITE CHAIN LINK SIZES
                                ;
```

UDA DM PROGRAM PARAMETERS

```
155         000401                  WBUFLN  =       257.                    ; WRITE BUFFER SIZE
156         000415                  RBUFLN  =       WBUFLN+12.              ; READ BUFFER SIZE
157         000007                  LINKLN  =       7.                      ; LINK SIZE
```

```
                                ;       XFC DEFINITION EQUATES
                                ;
000000                          BREAK   =       0.              ;BREAKPOINT XFC CODE
000001                          FORMAT  =       1.              ;FORMAT TRACK XFC CODE
000002                          XREAD   =       2.              ;READ N SECTORS XFC CODE
000003                          XWRITE  =       3.              ;WRITE N SECTORS XFC CODE
000004                          SEND    =       4.              ;SEND SDI COMMAND XFC CODE
000005                          RCV     =       5.              ;RECEIVE SDI MESSAGE XFC CODE
000006                          COMPARE =       6.              ;COMPARE DATA PATTERN TO BUFFER
000007                          STATUS  =       7.              ;RETURN DRIVE STATUS XFC CODE
000010                          ECHO    =       8.              ;ECHO DATA TO DRIVE XFC CODE
000011                          DINIT   =       9.              ;DRIVE INITIALIZE XFC CODE
000012                          WAITSI  =       10.             ;WAIT FOR SECTOR OR INDEX PULSE
000013                          UREAD   =       11.             ;READ UNIBUS MEMORY XFC CODE
000014                          UWRITE  =       12.             ;WRITE UNIBUS MEMORY XFC CODE
000015                          ECC     =       13.             ;DO ECC ON BUFFER XFC CODE
000016                          MRD     =       14.             ;SEND BUFFER TO MAINTENANCE READ COMMAND
000017                          MWR     =       15.             ;GET BUFFER FROM MAINTENANCE WRITE COMMAND
000020                          CVT     =       16.             ;CONVERT TO PHYSICAL ADDRESS XFC CODE
000021                          EXIT    =       17.             ;TERMINATE DM PROGRAM XFC CODE
                                ;
                                ;       GET STATUS OFFSETS
                                ;
000000                          ST.UNT  =       0.              ;UNIT NUMBER
000000                          ST.MSK  =       0.              ;SUBUNIT MASK
000001                          ST.STA  =       1.              ;STATUS BYTE
000001                          ST.MOD  =       1.              ;MODE BYTE
000002                          ST.ERR  =       2.              ;ERROR BYTE
000002                          ST.CON  =       2.              ;CONTROLLER BYTE
000002                          ST.C    =       2.              ;C BITS
000003                          ST.RTY  =       3.              ;RETRY COUNT/FAILURE CODE
                                ;
                                ;       STATUS BIT DEFINITIONS
                                ;
000200                          ST.OA   =       200             ; ONLINE TO ANOTHER (SET IF DRIVE UNAVAILABLE)
000100                          ST.RR   =       100             ; READJUSTMENT BIT (SET IF RECALIBRATION REQUIRED)
000040                          ST.DR   =       40              ; DIAGNOSTIC REQUEST (SET IF DIAGNOSTIC REQUESTED)
000020                          ST.SR   =       20              ; SPINDLE READY (SET IF SPINDLE READY)
000002                          ST.PS   =       2               ; PORT SWITCH (SET IF PORT SWITCH IN)
000001                          ST.RU   =       1               ; RUN/STOP SWITCH (SET IF RUN/STOP SWITCH IN)
000200                          ST.FE   =       200             ; FATAL ERROR (SET IF FATAL ERROR OCCURRED)
000100                          ST.RE   =       100             ; RETRIABLE ERROR (SET IF RETRIABLE ERROR OCCURRED)
000040                          ST.PE   =       40              ; PROTOCOL ERROR (SET IF PROTOCOL ERROR OCCURRED)
000020                          ST.DF   =       20              ; INITIALIZATION FAILURE (SET IF INIT FAILED)
000010                          ST.WE   =       10              ; WRITE ENABLE (SET IF WRT ATTEMPTED ON PROT DISK)
002000                          ST.FO   =       2000            ; FORMATTING (SET IF FORMATTING ENABLED)
001000                          ST.DB   =       1000            ; DIAGNOSTIC CYLS (SET IF DIAG CYL ACCESS ENABLED)
000400                          ST.S7   =       400             ; SECTOR SIZE (SET FOR 576 BYTE SECTORS)
```

```
                                ;       GET COMMON CHARACTERISTICS OFFSETS
                                ;
000000                          SHRTTO  =       0.              ;SHORT TIMEOUT <3:0>
000000                          SDIVER  =       0.              ;SDI VERSION <7:4>
000000                          XFERRT  =       0.              ;TRANSFER RATE <15:0>
000001                          LONGTO  =       1.              ;LONG TIMEOUT <3:0>
000001                          RETS    =       1.              ;RETRIES <7:4>
000001                          RCTCPS  =       1.              ;F/RCT COPIES <11:8>
000001                          SS      =       1.              ;SECTOR SIZE <15:15>
000002                          ERLEV   =       2.              ;ERROR RETRY LEVELS <7:0>
000002                          ECCRSH  =       2.              ;ECC THRESHOLD <15:8>
000003                          MICREV  =       3.              ;MICROCODE REVISION NUMBER <7:0>
000003                          HRDREV  =       3.              ;HARDWARE REVISION NUMBER <15:8>
000004                          DRVID   =       4.              ;UNIQUE DRIVE ID <47:0>
000007                          DRTYPE  =       7.              ;DRIVE TYPE IDENTIFIER <7:0>
000007                          REVS    =       7.              ;REVS/SECOND <15:8>
                                ;
                                ;       GET SUBUNIT CHARACTERISTICS OFFSETS
                                ;
                                ;THESE OFFSETS ARE CURRENTLY GIVEN AS FOLLOWING THE COMMON CHARACTERISTICS
                                ;
000013                          SUB     =       11.             ;OFFSET TO PUT SUBUNIT AFTER COMMON;
000000                          LBNCYL  =       0.              ;NUMBER OF CYLINDERS IN LBN AREA <31:0>
000001                          HICYL   =       1.              ;HI ORDER CYLINDER BITS <15:12>
000002                          GRPCYL  =       2.              ;GROUPS PER CYLINDER <7:0>
000002                          HILBN   =       2.              ;HI STARTING LBN <11:8>
000002                          HIXBN   =       2.              ;HI STARTING XBN <15:12>
000003                          TRKGRP  =       3.              ;TRACKS PER GROUP <7:0>
000003                          HIRBN   =       3.              ;HI STARTING RBN <11:8>
000003                          HIDBN   =       3.              ;HI STARTING DBN <15:12>
000004                          RBNTRK  =       4.              ;RBNS PER TRACK <6:0>
000004                          RM      =       4.              ;REMOVABLE MEDIA <7:7> 1=REMOVEABLE
000005                          DATPRE  =       5.              ;DATA PREAMBLE SIZE IN WORDS <7:0>
000005                          HDRPRE  =       5.              ;HEADER PREAMBLE SIZE IN WORDS <15:8>
000006                          MEDTYP  =       6.              ;MEDIA TYPE <31:0>
000010                          FCTSIZ  =       8.              ;FCT COPY SIZE <15:0>
000011                          LBNTRK  =       9.              ;LBNS PER TRACK <7:0>
000011                          GRPOFF  =       9.              ;GROUP OFFSET (SECTORS) <15:8>
000012                          LBNHST  =       10.             ;LBNS IN HOST AREA <31:0>
000014                          RCTCSZ  =       12.             ;RCT COPY SIZE <15:0>
000021                          XBNCYL  =       17.             ;CYLS IN XBN AREA <15:0>
000022                          DBNCYL  =       18.             ;CYLS IN DBN AREA <15:8>
                                ;
                                ;       UNIT CODES
                                ;
000001                          UNIT0   =       1.              ;UNIT ZERO CODE
000002                          UNIT1   =       2.              ;UNIT ONE CODE
000004                          UNIT2   =       4.              ;UNIT TWO CODE
000010                          UNIT3   =       8.              ;UNIT THREE CODE
                                ;
                                ;       BIT MASK DEFINITIONS
                                ;
                                ;
177400                          HIBYTE  =       177400          ;HIGH BYTE MASK
000377                          LOBYTE  =       000377          ;LOW BYTE MASK
007777                          HBHINB  =       7777            ;HI BYTE, HI NIBBLE MASK
170377                          HBLONB  =       170377          ;HI BYTE, LO NIBBLE MASK
```

```
     58          177417                  LBHINB  =       177417          ;LO BYTE, HI NIBBLE MASK
     59          177760                  LBLONB  =       177760          ;LO BYTE, LO NIBBLE MASK
     60                                  ;
     61          000001                  TIMEOUT =       1.              ;DRIVE TIMEOUT CODE
     62          000002                  HEADER  =       2.              ;HEADER COMPARE FAILURE CODE
     63          000004                  REVECT  =       4.              ;REVECTOR NEEDED CODE
     64                                  ;
     65          000002                  WRONG   =       2.              ;FIRST WORD NOT START FRAME CODE
     66          000004                  FRAME   =       4.              ;FRAMING ERROR CODE
     67          000010                  CHECK   =       8.              ;CHECKSUM ERROR CODE
     68                                  ;
     69          000001                  TOOBIG  =       1.              ;NUMBER OF WORDS EXCEEDS 7064
     70          000002                  LOW     =       2.              ;DM BUFFER ADDRESS IS LESS THAN 714
     71                                  ;
     72                                  ;       BIT MASK
     73                                  ;
     74          000001                  LARGE   =       1.              ;BLOCK NUMBER TOO LARGE
     75          000002                  OVERFL  =       2.              ;SECTOR NUMBER LARGER THAN 16 BITS
```

```
                        ;MAINTANENCE READ/WRITE REQUEST NUMBERS
                        ;
060000          T1MSIZ  =       0.+DU.SPC       ;GET FREE MEMORY PARAMETERS
060001          T2DLL   =       1.+DU.SPC       ;DOWNLINE LOAD DRIVE DIAGNOSTIC
060002          T2CMD   =       2.+DU.SPC       ;MANUAL INTERVENTION TEST 2 PROTOCOL
060003          T4MPRM  =       3.+DU.SPC       ;GET MASTER PARAMETERS FROM SW QUESTIONS
060004          T4UPRM  =       4.+DU.SPC       ;GET UNIT PARAMETERS FROM HW QUESTIONS
060005          T4BB1   =       5.+DU.SPC       ;GET BAD BLOCKS (1 THRU 14)
060006          T4BB2   =       6.+DU.SPC       ;GET REST OF BAD BLOCKS (15 AND 16)
060007          T4SOFT  =       7.+DU.SPC       ;ADD TO SOFT ERROR AND ECC COUNT
060010          T4SEEK  =       8.+DU.SPC       ;ADD 1 TO SEEK COUNT
060011          T4MXFR  =       9.+DU.SPC       ;ADD TO MEGABITS READ AND WRITTEN
060012          UTOTST  =       10.+DU.SPC      ;GET UNITS TO TEST
060013          ERRMES  =       11.+DU.SPC      ;PRINT ERROR MESSAGE
060014          ERRMC   =       12.+DU.SPC      ;TEST 4 ERROR REPORTING
060015          MESSAG  =       13.+DU.SPC      ;INFORMATION MESSAGE
060016          DONE    =       14.+DU.SPC      ;MARK DM PROGRAM AS NO LONGER RUNNING
                        ;
                        ;       OTHER BIT DEFINITIIONS
                        ;
000001          RCVRDY  =       1               ; RECIEVER READY 1 = READY
000002          ATTN    =       2               ; ATTENTION BIT FOR RETURN DRIVE SIGNALS XFC
000004          RCVERR  =       4               ; RECIEVER ERROR
000100          AVAIL   =       100             ; AVAILABLE 1 = AVAILABLE
000400          XMTERR  =       400             ; TRANSMIT ERROR
100000          RWRDY   =       100000          ; IF SET, UDA IS ABLE TO READ AND/OR WRITE TO DRIVE
                        ;
                        ;       SDI COMMANDS AND RESPONSES
                        ;
000204          DISCON  =       204             ; DISCONNECT DRIVE
000006          ERECOV  =       6               ; ERROR RECOVERY
000201          CHGMOD  =       201             ; CHANGE MODE
000213          DRVONL  =       213             ; DRIVE ONLINE
000014          DRVRUN  =       14              ; DRIVE RUN
000005          DRVCLR  =       5               ; DRIVE CLEAR OPCODE
000207          GETCHR  =       207             ; GET CHARACTERISTICS
000210          GETSUB  =       210             ; GET SUBUNIT CHARACTERISTICS
000011          GETSTA  =       11              ; GET STATUS
000216          IRECLB  =       216             ; RECALIBRATE
000012          INSEEK  =       12              ; INITIATE SEEK
000176          COMPLT  =       176             ; SUCCESSFUL COMPLETION
000175          UNSSUC  =       175             ; UNSUCCESSFUL COMPLETION
000170          CHRRES  =       170             ; GET CHARACTERISTICS RESPONSE
000167          SBCRES  =       167             ; GET SUBUNIT CHARACTERISTICS RESPONSE
000366          STSRES  =       366             ; GET STATUS RESPONSE
000350          ECHOC   =       350             ; DIAGNOSTIC ECHO COMMAND AND RESPONSE
                        ;
                        ;       ERROR CODES
                        ;
000000          FTLSYS  =             0         ; SYSTEM FATAL ERROR
040000          FTLDEV  =         40000         ; DEVICE FATAL
100000          ERHARD  =        100000         ; HARD ERROR
140000          ERSOFT  =        140000         ; SOFT ERROR
```

```
                                ;       DUMMY SDI CONTROL BLOCK OFFSETS
                                ;
000001                          D.LIMT  =       1                       ; DUMMY SDI SEARCH LIMIT
000002                          D.SCHR  =       2                       ; DUMMY POINTER TO SUBUNIT CHAR-5
                                ;
                                ;       DUP MESSAGE TYPES
010000                          DU.QUE = 10000
020000                          DU.DFL = 20000
030000                          DU.INF = 30000
040000                          DU.TER = 40000
050000                          DU.FTL = 50000
060000                          DU.SPC = 60000
                                .SBTTL  MACRO DEFINITIONS
                                ;
                                ;       MESSAGE CONTROL TABLE MACRO
                                ;
                                .MACRO MSG CMDBUF,CMDSZ,RPLBUF,RPLSZ,SUCCOM
                                .WORD CMDBUF                    ;ADDRESS OF COMMAND
                                .WORD CMDSZ                     ;SIZE OF COMMAND IN BYTES
                                .WORD RPLBUF                    ;ADDRESS OF REPLY
                                .WORD RPLSZ                     ;SIZE OF REPLY IN WORDS
                                .IF NB NUMBER
                                .WORD SUCCOM                    ; SUCCESSFUL COMPLETION CODE
                                .ENDC
                                .ENDM
```

MACRO DEFINITIONS

```
      1                                   .MACRO  BCS     LAB..
      2                                                           BCC     .+2
      3                                                           BR      LAB..
      4                                   .ENDM
```

```
                        ;       PUSH REGISTER MACRO
                        ;
                                .MACRO PUSH R9
                                .IRP X,<R9>
                                                                MOV X,-(SP)
                                .ENDR
                                .ENDM
                        ;
                        ;       POP REGISTER MACRO
                        ;
                                .MACRO POP R9
                                .IRP X,<R9>
                                                                MOV (SP)+,X
                                .ENDR
                                .ENDM
```

```
                        ;ERROR MACROS
                        ;THESE MACROS ARE CALLED TO REPORT ERRORS TO THE HOST PROGRAM.
                        ;THE MACRO NAMES ARE : ERRSF, ERRDF, ERRHRD, ERRSFT. EACH RESULTS IN THE HOST
                        ;BEING REQUESTED TO REPORT THE ERROR.
                        ;ARGUMENTS:     1 (MS$) MESSAGE POINTER
                        ;               2 (P1$) PARAMETER #1
                        ;               3 (P2$) PARAMETER #2
                        ;               4 (P3$) PARAMETER #3
                        ;               5 (P4$) PARAMETER #4
                        ;               6 (P5$) PARAMETER #5
                        ;               7 (P6$) PARAMETER #6
                        ;               8 (P7$) PARAMETER #7
                        ;               9 (P8$) PARAMETER #8
                        ;
                        ;THE MESSAGE POINTER MUST POINT TO AN ADDRESS IN THE OVERLAY 'MS' IMMEDIATELY
                        ;FOLLOWING THE MAIN CODE. ANY ADDRESS MODE MAY BE USED (E.G. #MS1, @R2).
                        ;THE ADDRESS MUST CONTAIN AN ASCII FORMAT STRING TO DETERMINE THE MESSAGE
                        ;TO PRINT.
                        ;THE PARAMETER ARGUMENTS ARE OPTIONAL. THEY SHOULD BE SUPPLIED ONLY WHEN
                        ;THERE IS DATA TO BE PASSED TO THE HOST THAT WILL BE USED IN PRINTING THE
                        ;MESSAGE. THESE PARAMETER ARGUMENTS ARE THE ADDRESS OF DATA TO BE PASSED
                        ;USING ANY ADDRESSING MODE DESIRED.
                        ;ALL REGISTERS ARE RETURNED UNCHANGED. IT SHOULD BE NOTED THAT ARGUMENTS
                        ;CONTAINING SOMETHING OTHER THAN A REGISTER NAME (E.G. #100 OR MEMADR)
                        ;ASSEMBLE TO INSTRUCTIONS THAT SAVE AND RESTORE A REGISTER ON THE STACK.

                                .MACRO ERRSF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ FTLSYS,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRDF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ FTLDEV,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRHRD MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NLIST
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ ERHARD,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .LIST
                                .ENDM

                                .MACRO ERRSFT MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ ERSOFT,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM
```

```
                        ;THE FOLLOWING MACRO ACTUALLY PROCESSES THE ERROR CALL TO THE HOST PROGRAM

                .MACRO ERROR$ ET$,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,ERRN$
                  .RADIX 8
                  PRMS=ARGS$-1
                  .IF LT,<PRMS>
                    .ERROR;NOT ENOUGH ARGUMENTS IN ERROR CALL
                  .ENDC
                  REGS$=-1
                  .IIF GE,<PRMS-8.>,PARGS. P8$
                  .IIF GE,<PRMS-7.>,PARGS. P7$
                  .IIF GE,<PRMS-6.>,PARGS. P6$
                  .IIF GE,<PRMS-5.>,PARGS. P5$
                  .IIF GE,<PRMS-4.>,PARGS. P4$
                  .IIF GE,<PRMS-3.>,PARGS. P3$
                  .IIF GE,<PRMS-2.>,PARGS. P2$
                  .IIF GE,<PRMS-1.>,PARGS. P1$
                  .IF GE REGS$
                    RSTR$ \REGS$
                  .ENDC
                  .RADIX 10
                  .LIST
                                                                CALL RERROR     ;ERROR # ERRN$'.
                  .NLIST
                  .RADIX 8
                  .LIST
                                                                .WORD ET$+ERRN
                                                                .WORD <PRMS*10000>+MS$
                  .NLIST
                  ERRN=ERRN+1
                .ENDM

                .MACRO PARGS.,ADDR$
                  .NTYPE PTYPE$,ADDR$
                  .IF EQ,<PTYPE$&70>
                    .IIF EQ,<PTYPE$&7>-REGS$,RSTR$ \REGS$
                    .LIST
                                                                MOV ADDR$,-(SP)
                    .NLIST
                  .IFF
                    .IF EQ,<PTYPE$&7>-1         ;PICK A REGISTER TO USE
                      REGU$=2                   ;SELECT R2 IF R1 IS USED IN PARAMETER FETCH
                    .IFF
                      REGU$=1                   ;OTHERWISE USE R1
                    .ENDC
                    .IF NE,<REGU$-REGS$>        ;IF REGISTER NOT ALREADY SAVED
                      .IF GE,REGS$
                        RSTR$ \REGS$            ;RESTORE CURRENT SAVED REGISTER
                      .ENDC
                      SAVR$ \REGU$              ;THEN SAVE SELECTED REGISTER
                    .ENDC
                    GETP$ \REGS$,ADDR$
                  .ENDC
                .ENDM

                .MACRO  DEVFTL  NUM,ARGS
                ERROR   FTLDEV,NUM,<ARGS>
```

```
                                                                MOV     #ERRMC,OUT.RQ
                        .ENDM
                        .MACRO  ERROR   TYPE,NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       5
                        MOVMSG  #ER'NUM,4
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC
                                                                MOV     #NUM!TYPE,R2
                                                                MOV     R2,OUT.02
                                                                MOV     #.,OUT.01
                        .RADIX
                        .ENDM
```

MACRO DEFINITIONS

```
                        .MACRO SAVR$ REGN
                          .LIST
                                                        MOV R'REGN,SAVREG
                          .NLIST
                          REGS$=REGN
                        .ENDM

                        .MACRO RSTR$ REGN
                          .LIST
                                                        MOV SAVREG,R'REGN
                          .NLIST
                          REGS$=-1
                        .ENDM

                        .MACRO GETP$ REGN,ADDR$
                          .LIST
                                                        MOV ADDR$,R'REGN
                                                        MOV R'REGN,-(SP)
                          .NLIST
                        .ENDM
```

MACRO DEFINITIONS

```
                        .MACRO  MSSGE   NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       3
                                                        MOV     LUNIT,OUT.01
                        MOVMSG  #'NUM,2
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC
                                                        PUSH    <R0,R1>
                                                        MOV     #MESSAG,R0
                                                        CALL    HOSTRQ
                                                        POP     <R1,R0>
                        .RADIX
                        .ENDM

                        .MACRO  MOVMSG  ARG,INDX

                        .IF     LT,INDX-10
                                                        MOV     ARG,OUT.0'INDX
                        .IFF
                                                        MOV     ARG,OUT.'INDX
                        .ENDC
                        .ENDM
```

MACRO DEFINITIONS

```
      1                                 ;ASSUME MACRO
      2                                         .MACRO  ASSUME  P1,P2
      3                                         .IF     NE,P1-P2
      4                                         .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
      5                                         .ENDC
      6                                         .ENDM
```

MACRO DEFINITIONS

```
                        ;RETURN DRIVE STATUS MACRO WITH ERROR REPORTING

                                .MACRO  DSTAT,LAB$,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   ME
                                .LIST
                                CALL    RDSTAT                  ; GET DRIVE STATUS
                                BIT     #10000,R1               ; SEE IF ANY ERRORS
                                BEQ     2$                      ; IF NO ERROR, BRANCH
                                BIT     #4000,R1                ; SEE IF XMIT ERROR
                                BEQ     1$                      ; IF SO, BRANCH
                                ERRHRD  E1                      ; REPORT INVALID STATUS ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        1$:     ERRHRD  E2                      ; REPORT XMIT ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        2$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;
                        ;       XOR THE CONTENTS OF TWO REGISTERS
                        ;
                        .MACRO  RXOR    REG1,REG2
                        MOV     REG2,-(SP)              ; SAVE REGISTER REG2
                        BIC     REG1,REG2               ; CLEAR COORESPONDING BITS IN REG2
                        BIC     (SP)+,REG1              ; CLEAR COORESPONDING BITS IN REG1
                        BIS     REG1,REG2               ; OR WHAT'S LEFT
                        .ENDM
```

MACRO DEFINITIONS

```
                        ;
                        ;       CONVERT .BLKW CALLS TO ACTUAL WORDS GENERATED
                        ;
                        .MACRO  .BLKW   COUNT
                        .NLIST
                        .REPT   COUNT
                        .WORD   0
                        .ENDR
                        .LIST
                        .ENDM
```

MACRO DEFINITIONS

```
                        ;       SDI INTERCHANGE WITH DRIVE WITH ERROR REPORTING
                        ;
                                .MACRO  TALKX   ERRLAB,Ei,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   ME
                                .LIST
                                CALL    TALKER                  ; INITIATE SDI INTERCHANGE
                                TST     R3                      ; SEE IF ERROR OCCURRED
                                BEQ     12$                     ; IF NOT, BRANCH
                                BPL     11$                     ; IF SO, BRANCH
                                ERRHRD  E1;SEND COMMAND ERROR
                                BR      ERRLAB
                        11$:    ERRHRD  E2;RECEIVE COMMAND ERROR
                                BR      ERRLAB
                        12$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

START OF TEST CODE

```
      1                                 .SBTTL  START OF TEST CODE
      2                                 ;THE FOLLOWING IS FOR DEBUG PURPOSES ONLY. CAN BE NOP OR BREAKPOINT.
      3
      4 001364  114007                          CLR R0                          ;CHANGE TO BREAKPOINT FOR DEBUG
      5
      6                                 ;INITIALIZE STACK
      7
      8 001365  104206  002320                  MOV #STACK,SP                   ;SET UP STACK POINTER
     18 001367                                  BR      START                   ; BRANCH OVER SUPPORT CODE
        001367  000000  002321                  ^O0,START
```

```
                                        .SBTTL  PROGRAM VARIABLES
                                ;PROGRAM VARIABLES
                                ;
                                ;UNIT NUMBER STORAGE FOR DISK DRIVES TO TEST
                                ;
001371  177777  177777  177777  UNITS:  .WORD   -1,-1,-1,-1            ;LOGICAL UNIT NUMBER IF POSITIVE
001374  177777
001375  177777  177777  177777          .WORD   -1,-1,-1,-1           ;DISK DRIVE NOT TO BE TESTED IF NEGATIVE
001400  177777
001401  177777  177777  177777          .WORD   -1,-1,-1,-1
001404  177777
001405  177777  177777  177777          .WORD   -1,-1,-1,-1
001410  177777
                                ;
001411  000000                  UNITNB: .WORD   0                       ;NUMBER OF UNIT CURRENTLY UNDER TEST
                                                                        ;POINTER INTO TABLE ABOVE
                                ;
001412  000000                  SDI:    .WORD   0                       ;SDI INTERCONNECT CODE FOR XFC CALLS
                                ;
001413  000000                  SAVSTA: .WORD   0                       ;SAVE STATUS
001414  000000                  SAVRID: .WORD   0                       ;SAVE REGION ID
                                ;
                                ;DRIVE RESPONSE BUFFERS
        000200                  STSIZE  =       200
001415                          ST:     .BLKW   200
                                ;
                                ;
001615  000000                  DIAG.1: .WORD   0                       ; = 0, 1ST DIAGNOSE COMMAND
                                                                        ; NOT = 0, DRIVE TESTED BEFORE
001616  000000                  NAM.1:  .WORD   0                       ;HOLD PROGRAM NAME
001617  000000                  NAM.2:  .WORD   0
001620  000000                  NAM.3:  .WORD   0
001621  000000                          .WORD   0
                                ;
001622  000000                  SENDHR: .WORD   0                       ;IF 0, DON'T SEND HOSTRQ IN TALKER
                                                                        ;IF NOT 0, SEND HOSTRQ -> DOING DIAGNOSE COMMAND
                                ;
001623  177777                  LUNIT:  .WORD -1                        ;LOGICAL UNIT NUMBER (-1 FOR NOT AVAILABLE)
                                ;
001624  000000                  SAVREG: .WORD   0                       ;STORAGE FOR REGISTER AT CALL TIME
001625  000012                  SDISTO: .WORD   10.                     ; SDI SHORT TIMEOUT
001626  000024                  SDILTO: .WORD   20.                     ; SDI LONG TIMEOUT
001627  000000                  SAFWRD: .WORD   0                       ; SAFE WORD TO USE IN DO.DCL
                                ;
001630  004212                  SER18E: .WORD   SER18A                  ; FOR MULTIPLE SUBUNIT ERROR REPORTING
001631  004206                          .WORD   SER18B
001632  004202                          .WORD   SER18C
001633  004176                          .WORD   SER18D
                                ;
        001750                  MAXSND  =       1000.                   ; MAXIMUM # OF SENDS
                                ;
        000017                  WRM.OP  =       17                      ; WRITE MEMORY COMMAND
        000215                  RDM.OP  =       215                     ; READ MEMORY COMMAND
        000003                  DIA.OP  =       3                       ; DIAGNOSE COMMAND
        000207                  GETCHR  =       207                     ; GET COMMON CHARACTERISTICS COMMAND
        000162                  RDM.EN  =       162                     ; READ MEMORY RESPONSE
        000374                  DIA.EN  =       374                     ; DIAGNOSE RESPONSE
```

```
     54          000170                  GCC.EN  =       170             ; GET COMMON CHARACTERISTICS
     55                                  ;
     56                                  ;       DIAGNOSE ET AND DA
     57                                  ;
     58          100000                  ERRTYP  =       100000          ;ERROR TYPE ET
     59          040000                  DATAVL  =        40000          ;DATA AVAILABLE
     60                                  ;
     61                                  ;       REGION ID'S
     62                                  ;
     63          177774                  FFFC    =       FFFD - 1
     64          177775                  FFFD    =       177775
     65          177776                  FFFE    =       FFFD + 1
     66
```

```
                                        .SBTTL  SDI COMMANDS USED FOR TEST 2
001634    377     350       ECHOD:    .BYTE   377,ECHOC               ;ECHO DATA TO SEND TO DRIVE
001635    000     350                 .BYTE   000,ECHOC
001636    252     350                 .BYTE   252,ECHOC
001637    360     350                 .BYTE   360,ECHOC
001640    017     350                 .BYTE   017,ECHOC
001641  000000                        .WORD   0                       ;END MARKER

                                        .SBTTL  COMMAND BUFFERS FOR SEND XFC
                                    ;COMMAND BUFFERS FOR SEND XFC

001642                              CR.GST: MSG     GETST,1,ST,7            ; GET STATUS SDI INFORMATION
001642  001647                              .WORD GETST                     ;ADDRESS OF COMMAND
001643  000001                              .WORD 1                 ;SIZE OF COMMAND IN BYTES
001644  001415                              .WORD ST                        ;ADDRESS OF REPLY
001645  000007                              .WORD 7                 ;SIZE OF REPLY IN WORDS
001646  000000                              .WORD                   ; SUCCESSFUL COMPLETION CODE
001647    000     011       GETST:    .BYTE   0,GETSTA
                                    ;
001650                              CR.DRC: MSG     DRC,2,ST,7              ; DRIVE CLEAR SDI INFORMATION
001650  001655                              .WORD DRC                       ;ADDRESS OF COMMAND
001651  000002                              .WORD 2                 ;SIZE OF COMMAND IN BYTES
001652  001415                              .WORD ST                        ;ADDRESS OF REPLY
001653  000007                              .WORD 7                 ;SIZE OF REPLY IN WORDS
001654  000000                              .WORD                   ; SUCCESSFUL COMPLETION CODE
001655    000     005       DRC:      .BYTE   0,DRVCLR
001656  177777              DRCBYT:   .WORD   177777
                                    ;
001657                              CR.WRM: MSG     WRM,7,ST,7              ; MEMORY WRITE COMMAND INFO
001657  001664                              .WORD WRM                       ;ADDRESS OF COMMAND
001660  000007                              .WORD 7                 ;SIZE OF COMMAND IN BYTES
001661  001415                              .WORD ST                        ;ADDRESS OF REPLY
001662  000007                              .WORD 7                 ;SIZE OF REPLY IN WORDS
001663  000000                              .WORD                   ; SUCCESSFUL COMPLETION CODE
001664     000      017          WRM:      .BYTE   0,WRM.OP
001665  000000  000000  000000             .WORD   0,0,0
001670  000000  000000  000000             .WORD   0,0,0
001673              BUF1:     .BLKW   200
                                    ;
002073                              CR.RDM: MSG     RDM,6,ST,STSIZE         ; MEMORY READ COMMAND INFO
002073  002100                              .WORD RDM                       ;ADDRESS OF COMMAND
002074  000006                              .WORD 6                 ;SIZE OF COMMAND IN BYTES
002075  001415                              .WORD ST                        ;ADDRESS OF REPLY
002076  000200                              .WORD STSIZE                    ;SIZE OF REPLY IN WORDS
002077  000000                              .WORD                   ; SUCCESSFUL COMPLETION CODE
002100     000      215          RDM:      .BYTE   0,RDM.OP
002101  000000  000000  000000             .WORD   0,0,0,0
002104  000000
002105  000000  000000  000000             .WORD   0,0,0,0
002110  000000
                                    ;
002111                              CR.GCR: MSG     GCR,1,ST,12.            ; GET CHARACTERISTICS
002111  002116                              .WORD GCR                       ;ADDRESS OF COMMAND
002112  000001                              .WORD 1                 ,SIZE OF COMMAND IN BYTES
002113  001415                              .WORD ST                        ;ADDRESS OF REPLY
002114  000014                              .WORD 12.                       ;SIZE OF REPLY IN WORDS
002115  000000                              .WORD                   ; SUCCESSFUL COMPLETION CODE
```

COMMAND BUFFERS FOR SEND XFC

```
     31 002116     000     207          GCR:    .BYTE   0,GETCHR                ; GET CHARACTERISTICS
     32                                 ;
     33 002117                          CR.ONL: MSG     ONL,2,ST,7              ; ONLINE COMMAND
        002117  002124                          .WORD ONL                       ;ADDRESS OF COMMAND
        002120  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
        002121  001415                          .WORD ST                        ;ADDRESS OF REPLY
        002122  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
        002123  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
     34 002124     000     213          ONL:    .BYTE   0,213
     35 002125  000077                          .WORD   77
     36                                 ;
     37 002126                          LONG:                                   ; ALL LONG TIMEOUT COMMANDS FOLLOW
     38                                 ;
     39 002126                          CR.DIA: MSG     DIA,3,ST,7              ; DIAGNOSE COMMAND INFORMATION
        002126  002133                          .WORD DIA                       ;ADDRESS OF COMMAND
        002127  000003                          .WORD 3                 ;SIZE OF COMMAND IN BYTES
        002130  001415                          .WORD ST                        ;ADDRESS OF REPLY
        002131  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
        002132  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
     40 002133     000     003          DIA:    .BYTE   0,DIA.OP
     41 002134  000000                          .WORD   0
     42 002135  000000                          .WORD   0
     43 002136                          CR.RUN: MSG     RUN,1,ST,7
        002136  002143                          .WORD RUN                       ;ADDRESS OF COMMAND
        002137  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
        002140  001415                          .WORD ST                        ;ADDRESS OF REPLY
        002141  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
        002142  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
     44 002143     000     014          RUN:    .BYTE   0,DRVRUN
     45 002144                          CR.INR: MSG     INR,1,ST,7
        002144  002151                          .WORD INR                       ;ADDRESS OF COMMAND
        002145  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
        002146  001415                          .WORD ST                        ;ADDRESS OF REPLY
        002147  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
        002150  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
     46 002151     000     216          INR:    .BYTE   0,IRECLB
```

```
STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS

                                        .SBTTL  STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS
                                ;STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS

                                ;OUT BUFFER - DATA TO SEND TO HOST

002152  000000                  OUT.RQ: .WORD 0                 ;HOST REQUEST CODE
002153  000000                  OUT.01: .WORD 0                 ;DATA ARGUMENT  1
002154  000000                  OUT.02: .WORD 0                 ;DATA ARGUMENT  2
002155  000000                  OUT.03: .WORD 0                 ;DATA ARGUMENT  3
002156  000000                  OUT.04: .WORD 0                 ;DATA ARGUMENT  4
002157  000000                  OUT.05: .WORD 0                 ;DATA ARGUMENT  5
002160  000000                  OUT.06: .WORD 0                 ;DATA ARGUMENT  6
002161  000000                  OUT.07: .WORD 0                 ;DATA ARGUMENT  7
002162  000000                  OUT.08: .WORD 0                 ;DATA ARGUMENT  8
002163  000000                  OUT.09: .WORD 0                 ;DATA ARGUMENT  9
002164  000000                  OUT.10: .WORD 0                 ;DATA ARGUMENT 10
002165  000000                  OUT.11: .WORD 0                 ;DATA ARGUMENT 11
002166  000000                  OUT.12: .WORD 0                 ;DATA ARGUMENT 12
002167  000000                  OUT.13: .WORD 0                 ;DATA ARGUMENT 13
002170  000000                  OUT.14: .WORD 0                 ;DATA ARGUMENT 14
002171  000000                  OUT.15: .WORD 0                 ;DATA ARGUMENT 15
002172  000000                  OUT.16: .WORD 0                 ;DATA ARGUMENT 16
002173  000000                  OUT.17: .WORD 0                 ;DATA ARGUMENT 17
002174  000000                  OUT.18: .WORD 0                 ;DATA ARGUMENT 18
002175  000000                  OUT.19: .WORD 0                 ;DATA ARGUMENT 19
002176  000000                  OUT.20: .WORD 0                 ;DATA ARGUMENT 20
002177  000000                  OUT.21: .WORD 0                 ;DATA ARGUMENT 21
002200  000000                  OUT.22: .WORD 0                 ;DATA ARGUMENT 22
002201  000000                  OUT.23: .WORD 0                 ;DATA ARGUMENT 23
002202  000000                  OUT.24: .WORD 0                 ;DATA ARGUMENT 24
002203  000000                  OUT.25: .WORD 0                 ;DATA ARGUMENT 25
002204  000000                  OUT.26: .WORD 0                 ;DATA ARGUMENT 26
002205  000000                  OUT.27: .WORD 0                 ;DATA ARGUMENT 27
002206  000000                  OUT.28: .WORD 0                 ;DATA ARGUMENT 28
002207  000000                  OUT.29: .WORD 0                 ;DATA ARGUMENT 29
002210  000000                  OUT.30: .WORD 0                 ;DATA ARGUMENT 30
002211  000000                  OUT.31: .WORD 0                 ;DATA ARGUMENT 31
002212  000000                  OUT.32: .WORD 0                 ;DATA ARGUMENT 32
002213  000000                  OUT.33: .WORD 0                 ;DATA ARGUMENT 33
002214  000000                  OUT.34: .WORD 0                 ;DATA ARGUMENT 34

                                ;IN BUFFER - DATA RECEIVED FROM HOST

002215  000000                  IN.RQ:  .WORD 0                 ;HOST REQUEST CODE (ECHO)
002216  000000                  IN.01:  .WORD 0                 ;DATA ARGUMENT  1
002217  000000                  IN.02:  .WORD 0                 ;DATA ARGUMENT  2
002220  000000                  IN.03:  .WORD 0                 ;DATA ARGUMENT  3
002221  000000                  IN.04:  .WORD 0                 ;DATA ARGUMENT  4
002222  000000                  IN.05:  .WORD 0                 ;DATA ARGUMENT  5
002223  000000                  IN.06:  .WORD 0                 ;DATA ARGUMENT  6
002224  000000                  IN.07:  .WORD 0                 ;DATA ARGUMENT  7
002225  000000                  IN.08:  .WORD 0                 ;DATA ARGUMENT  8
002226  000000                  IN.09:  .WORD 0                 ;DATA ARGUMENT  9
002227  000000                  IN.10:  .WORD 0                 ;DATA ARGUMENT 10
002230  000000                  IN.11:  .WORD 0                 ;DATA ARGUMENT 11
002231  000000                  IN.12:  .WORD 0                 ;DATA ARGUMENT 12
002232  000000                  IN.13:  .WORD 0                 ;DATA ARGUMENT 13
```

```
     58 002233  000000                IN.14:  .WORD 0                 ;DATA ARGUMENT 14
     59 002234  000000                IN.15:  .WORD 0                 ;DATA ARGUMENT 15
     60 002235  000000                IN.16:  .WORD 0                 ;DATA ARGUMENT 16
     61 002236  000000                IN.17:  .WORD 0                 ;DATA ARGUMENT 17
     62 002237  000000                IN.18:  .WORD 0                 ;DATA ARGUMENT 18
     63 002240  000000                IN.19:  .WORD 0                 ;DATA ARGUMENT 19
     64 002241  000000                IN.20:  .WORD 0                 ;DATA ARGUMENT 20
     65 002242  000000                IN.21:  .WORD 0                 ;DATA ARGUMENT 21
     66 002243  000000                IN.22:  .WORD 0                 ;DATA ARGUMENT 22
     67 002244  000000                IN.23:  .WORD 0                 ;DATA ARGUMENT 23
     68 002245  000000                IN.24:  .WORD 0                 ;DATA ARGUMENT 24
     69 002246  000000                IN.25:  .WORD 0                 ;DATA ARGUMENT 25
     70 002247  000000                IN.26:  .WORD 0                 ;DATA ARGUMENT 26
     71 002250  000000                IN.27:  .WORD 0                 ;DATA ARGUMENT 27
     72 002251  000000                IN.28:  .WORD 0                 ;DATA ARGUMENT 28
     73 002252  000000                IN.29:  .WORD 0                 ;DATA ARGUMENT 29
     74 002253  000000                IN.30:  .WORD 0                 ;DATA ARGUMENT 30
     75 002254  000000                IN.31:  .WORD 0                 ;DATA ARGUMENT 31
     76 002255  000000                IN.32:  .WORD 0                 ;DATA ARGUMENT 32
     77 002256  000000                IN.33:  .WORD 0                 ;DATA ARGUMENT 33
     78 002257  000000                IN.34:  .WORD 0                 ;DATA ARGUMENT 34
     79         000043                BUFSIZ  =       . - IN.RQ       ;SIZE OF BUFFER
```

```
                                        .SBTTL  STACK AREA
                                ;STACK AREA
                                
002260  123456                          .WORD 123456            ;END MARKER FOR STACK
002261                                  .BLKW 31.               ;STACK
002320  123456                  STACK:  .WORD 123456            ;MARKER FOR STACK UNDERFLOW
```

```
                                .SBTTL  GETU   - POLL ALL PORTS, THEN GET UNITS TO TEST

                                ;GET WHICH UNITS TO TEST

002321                          START:
                                ;
                                ;       GET THE UNITS TO TEST
                                ;
                                ;
                                ;       POLL ALL PORTS AND FILL IN A UDA PORT INFORMATION TABLE (UNITS)
                                ;
002321  104205  000001                  MOV     #1,R5           ; MOVE INITIAL MASK TO R5
002323  104204  001371                  MOV     #UNITS,R4       ; R4 POINTS TO UNIT TABLE
002325                          5$:     PUSH    R4              ; SAVE R4
002325  100464                                                                  MOV R4,-(SP)
002326  104052                          MOV     R5,R2           ; MOVE MASK TO R2
002327                                  CALL    RDSTAT          ; GET DRIVE'S STATUS
002327  020000  006027                  ^O020000,RDSTAT
002331  102201  010000                  BIT     #10000,R1       ; SEE IF ERROR
002333                                  BEQ     10$             ; IF NOT, BRANCH
002333  010000  002343                  ^O010000,10$
002335  104203  003772                  MOV     #SER10,R3       ; NO DRIVE ATTACHED
002337  100643  000001                  MOV     R3,1(R4)        ; SAVE ERROR MESSAGE
002341                                  BR      85$             ; REPORT
002341  000000  002563                  ^O0,85$
002343  114003                  10$:    CLR     R3              ; SET UP TIMEOUT COUNT
002344                          15$:    CALL    RDSTAT          ; GET STATUS
002344  020000  006027                  ^O020000,RDSTAT
002346  102201  000001                  BIT     #RCVRDY,R1      ; SEE IF RECEIVER READY ASSERTED
002350                                  BNE     20$             ; IF SO, BRANCH
002350  050000  002363                  ^O50000,20$
002352  117403                          DEC     R3              ; DECREMENT COUNT
002353                                  BNE     15$             ; IF INCOMPLETE, BRANCH
002353  050000  002344                  ^O50000,15$
002355  104203  004005                  MOV     #SER11,R3       ; RECEIVER READY NEVER ASSERTED
002357  100643  000001                  MOV     R3,1(R4)        ; SAVE ERROR MESSAGE
002361                                  BR      85$             ; REPORT
002361  000000  002563                  ^O0,85$
002363  102201  000100          20$:    BIT     #AVAIL,R1       ; SEE IF DRIVE IS AVAILABLE
002365                                  BNE     25$             ; IF SO, BRANCH
002365  050000  002375                  ^O50000,25$
002367  104203  004215                  MOV     #SER40,R3       ; GET SECONDARY ERROR
002371  100643  000001                  MOV     R3,1(R4)        ; SAVE
002373                                  BR      85$             ; EXIT
002373  000000  002563                  ^O0,85$
002375  104202  001750          25$:    MOV     #MAXSND,R2      ; SET UP MAXIMUM TRIES AT SENDING
002377  104203  001642                  MOV     #CR.GST,R3      ; R3 POINTS TO GET STATUS COMMAND
002401  104137                  30$:    MOV     (R3),R0         ; SET ADR OF SDI COMMAND BUFFER
002402  104631  000001                  MOV     1(R3),R1        ; SET BUFFER LENGTH
002404                                  PUSH    R2              ; SAVE R2
002404  100462                                                                  MOV R2,-(SP)
002405  104052                          MOV     R5,R2           ; SETUP FOR SEND
002406  060004                          XFC     SEND            ; SEND COMMAND
002407                                  POP     R2              ; RESTORE COUNT
002407  104262                                                                  MOV (SP)+,R2
002410  115001                          TST     R1              ; DID UNIT ACCEPT COMMAND
```

```
     46 002411
        002411  010000  002424                 BEQ     35$             ; IF SO, BRANCH
                                                ^010000,35$
     47 002413  117402                          DEC     R2              ; DECREMENT COUNT
     48 002414                                  BNE     30$             ; IF UNEXPIRED, BRANCH
        002414  050000  002401                  ^050000,30$
     49 002416  104203  004023                  MOV     #SER12,R3       ; GET ERROR NUMBER
     50 002420  100643  000001                  MOV     R3,1(R4)        ; SAVE
     51 002422                                  BR      85$
        002422  000000  002563                  ^00,85$
     52 002424                          35$:    PUSH    R4              ; SAVE R4
        002424  100464                                                  MOV R4,-(SP)
     53 002425  104204  000003                  MOV     #3,R4           ; SET UP SHORT TIMEOUT
     54 002427  104637  000002          40$:    MOV     2(R3),R0        ; SET DATA BUFFER ADDRESS
     55 002431  104631  000003                  MOV     3(R3),R1        ; SET BUFFER LENGTH
     56 002433  104052                          MOV     R5,R2           ; SETUP FOR RECEIVE
     57 002434  060005                          XFC     RCV             ; RECEIVE SDI COMMAND
     58 002435  115001                          TST     R1              ; DID ERROR OCCUR
     59 002436                                  BEQ     70$             ; IF NOT, BRANCH
        002436  010000  002511                  ^010000,70$
     60 002440  106201  000001                  CMP     #1,R1           ; SEE IF TIMEOUT
     61 002442                                  BNE     45$             ; IF NOT, BRANCH
        002442  050000  002454                  ^050000,45$
     62 002444  117404                          DEC     R4              ; DECREMENT TIMEOUT VALUE
     63 002445                                  BNE     40$             ; IF NOT TIMEOUT, BRANCH
        002445  050000  002427                  ^050000,40$
     64 002447                                  POP     R4              ; RESTORE R4
        002447  104264                                                  MOV (SP)+,R4
     65 002450  104203  004035                  MOV     #SER13,R3       ; GET ERROR NUMBER
     66 002452                                  BR      65$             ; BRANCH TO EXIT
        002452  000000  002505                  ^00,65$
     67 002454                          45$:    POP     R4              ; RESTORE R4
        002454  104264                                                  MOV (SP)+,R4
     68 002455  110601                          ROR     R1              ; ROTATE INTO POSITION TO TEST
     69 002456  110601                          ROR     R1              ; SEE IF FIRST WORD NOT START FRAME
     70 002457                                  BCC     50$             ; IF NOT, BRANCH
        002457  040000  002465                  ^040000,50$
     71 002461  104203  004050                  MOV     #SER14,R3       ; GET ERROR NUMBER
     72 002463                                  BR      65$             ; BRANCH TO END OF LOOP
        002463  000000  002505                  ^00,65$
     73 002465  110601                  50$:    ROR     R1              ; SEE IF FRAMING ERROR
     74 002466                                  BCC     55$             ; IF NOT, BRANCH
        002466  040000  002474                  ^040000,55$
     75 002470  104203  004076                  MOV     #SER15,R3       ; GET ERROR NUMBER
     76 002472                                  BR      65$             ; BRANCH TO END OF LOOP
        002472  000000  002505                  ^00,65$
     77 002474  110601                  55$:    ROR     R1              ; SEE IF CHECKSUM ERROR
     78 002475                                  BCC     60$             ; IF NOT, BRANCH
        002475  040000  002503                  ^040000,60$
     79 002477  104203  004120                  MOV     #SER16,R3       ; GET ERROR NUMBER
     80 002501                                  BR      65$             ; BRANCH TO END OF LOOP
        002501  000000  002505                  ^00,65$
     81 002503  104203  004143          60$:    MOV     #SER17,R3       ; GET ERROR NUMBER
     82 002505  100643  000001          65$:    MOV     R3,1(R4)        ; SAVE
     83 002507                                  BR      85$             ; ERANCH TO END OF LOOP
        002507  000000  002563                  ^00,85$
```

```
                                        ;
                                        ;       NOW FILL IN THE TABLE WITH ALL THE SUBUNIT NUMBERS
                                        ;
 002511                         70$:    POP     R4              ; RESTORE R4
 002511 104264                                                                  MOV (SP)+,R4
 002512 104207 177777                   MOV     #-1,R0          ; GET 'NO UNITS' FLAG
 002514 100647 000001                   MOV     R0,1(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
 002516 100647 000002                   MOV     R0,2(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
 002520 104307 001415                   MOV     ST,R0           ; R0 HAS UNIT NUMBER
 002522 104072                          MOV     R0,R2           ; COPY R0 TO R2
 002523 103207 170000                   BIC     #^CHBHINB,R0    ; R0 HAS UNIT NUMBER
 002525                                 PUSH    <R1,R2>         ; SAVE R1 AND R2
 002525 100461                                                                  MOV R1,-(SP)
 002526 100462                                                                  MOV R2,-(SP)
 002527 104052                          MOV     R5,R2           ; MOVE UDA PORT MASK TO R2
 002530                                 CALL    RDSTAT          ; GET STATUS
 002530 020000 006027                   ^O020000,RDSTAT
 002532 102201 000002                   BIT     #ATTN,R1        ; SEE IF SPINABLE
 002534                                 BNE     75$             ; IF SO, BRANCH
 002534 050000 002540                   ^O050000,75$
 002536 101207 010000                   BIS     #10000,R0       ; SET 'NOT SPINABLE' FLAG
 002540                         75$:    POP     <R2,R1>         ; RESTORE
 002540 104262                                                                  MOV (SP)+,R2
 002541 104261                                                                  MOV (SP)+,R1
 002542 101207 040000                   BIS     #40000,R0       ; FLAG UNIT AS NOT TESTED
 002544 110702                          SWAB    R2              ; SWAP R2'S BYTES
 002545 110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
 002546 110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
 002547 110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
 002550 110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
 002551 103202 177760                   BIC     #LBLONB,R2      ; CLEAR ALL BUT SUBUNIT BITS
 002553 100147                          MOV     R0,(R4)         ; MOVE UNIT NUMBER INTO UNITS
 002554 110602                  80$:    ROR     R2              ; MOVE SUBUNIT BIT TO CARRY
 002555                                 BCC     85$             ; IF NO MORE SUBUNITS, BRANCH
 002555 040000 002563                   ^O040000,85$
 002557 100247                          MOV     R0,(R4)+        ; MOVE SUBUNIT NUMBER TO TABLE
 002560 115407                          INC     R0              ; INCREMENT SUBUNIT NUMBER
 002561                                 BR      80$             ; BRANCH
 002561 000000 002554                   ^O00,80$

 002563                         85$:    POP     R4              ; R4 POINTS TO START OF UNIT JUST HANDLED
 002563 104264                                                                  MOV (SP)+,R4
 002564 105204 000004                   ADD     #4,R4           ; R4 WILL POINT TO NEXT UNIT (CLEAR CARRY FOR ROL)
 002566 110205                          ROL     R5              ; R5 HAS NEXT UNIT PORT MASK
 002567 106205 000020                   CMP     #20,R5          ; SEE IF ALL PORTS TESTED
 002571                                 BNE     5$              ; IF NOT, BRANCH
 002571 050000 002325                   ^O050000,5$
```

```
                                ;
                                ;       NOW GET THE PLUG NUMBERS TO TEST AND FIND THEM IN THE TABLE
                                ;
 002573  104207  060012               MOV     #UTOTST,R0      ; GET WHAT SUBUNIT NUMBERS TO TEST REQUEST
 002575                               CALL    HOSTRQ          ; GET THE PLUG NUMBERS
 002575  020000  006066               ^O020000,HOSTRQ
 002577  104207  002216               MOV     #IN.01,R0       ; R0 POINTS TO UNIT NUMBERS TO TEST
 002601  104201  001371       90$:    MOV     #UNITS,R1       ; R1 POINTS TO UDA PORT INFORMATION
 002603  104112               95$:    MOV     (R1),R2         ; R2 HAS UNIT
 002604                               BMI     115$            ; IF NONE, BRANCH
 002604  070000  002634               ^O070000,115$
 002606                               PUSH    R1              ; SAVE R1
 002606  100461                                                                       MOV R1,-(SP)
 002607  103202  170000       100$:   BIC     #170000,R2      ; CLEAR 'NOT TESTED', DO NOT LEAVE 'UNSPINABLE' SET
 002611  106172                       CMP     (R0),R2         ; SEE IF IT IS A UNIT TO TEST
 002612                               BNE     105$            ; NO MATCH
 002612  050000  002620               ^O050000,105$
 002614  100112                       MOV     R2,(R1)         ; SAVE UNIT AS ONE TO TEST
 002615                               POP     R1              ; RESTORE STACK
 002615  104261                                                                       MOV (SP)+,R1
 002616                               BR      160$            ; LOOK FOR THE NEXT ONE
 002616  000000  002754               ^O00,160$
 002620  115401       105$:   INC     R1              ; POINT TO NEXT SUBUNIT
 002621  104012                       MOV     R1,R2           ; COPY TO R2
 002622  107202  001371               SUB     #UNITS,R2       ; SUBTRACT STARTING ADDRESS
 002624  102202  000003               BIT     #3,R2           ; SEE IF STILL ON SAME UNIT
 002626                               BEQ     110$            ; IF NOT, BRANCH
 002626  010000  002633               ^O010000,110$
 002630  104112                       MOV     (R1),R2         ; SEE IF ANY MORE SUBUNITS
 002631                               BPL     100$            ; IF SO, BRANCH
 002631  030000  002607               ^O030000,100$
 002633       110$:   POP     R1              ; RESTORE R1
 002633  104261                                                                       MOV (SP)+,R1
 002634  105201  000004       115$:   ADD     #4,R1           ; LOOK AT NEXT UNIT
 002636  106201  001411               CMP     #UNITS+16.,R1   ; SEE IF ENTIRE TABLE SEARCHED
 002640                               BNE     95$             ; IF NOT, BRANCH
 002640  050000  002603               ^O050000,95$
```

```
                                ;
                                ;       DIDN'T FIND THE REQUESTED UNITS -- DUMP ALL KNOWLEDGE OF THE
                                ;       UDA PORTS AND DIE
                                ;
002642  104170  002155                MOV     (R0),OUT.03     ; SAVE UNIT NUMBER IN REQUEST BUFFER
002644  104204  002157                MOV     #OUT.05,R4      ; R4 POINTS TO OUTPUT BUFFER
002646  104205  001371                MOV     #UNITS,R5       ; R5 POINTS TO UNIT TABLE
002650  104157              120$:     MOV     (R5),R0         ; GET FIRST WORD OF UNIT
002651                            BPL     125$            ; IF VALID UNIT WAS FOUND, BRANCH
002651  030000  002660                ^O030000,125$
002653  104657  000001                MOV     1(R5),R0        ; GET POINTER TO ERROR MESSAGE
002655  100247                        MOV     R0,(R4)+        ; SAVE IN OUTPUT BUFFER
002656                            BR      155$            ; EXIT
002656  000000  002720                ^O00,155$
002660              125$:     PUSH    R5              ; SAVE POINTER TO UNIT TABLE
002660  100465                                                                    MOV R5,-(SP)
002661  102207  010000                BIT     #10000,R0       ; SEE IF DRIVE UNSPINABLE
002663                            BEQ     130$            ; IF SPINABLE, BRANCH
002663  010000  002671                ^O010000,130$
002665  104207  004237                MOV     #SER41,R0       ; REPORT DRIVE(S) UNSPINABLE
002667                            BR      135$            ; BRANCH
002667  000000  002673                ^O00,135$
002671  104207  004170      130$:     MOV     #SER18,R0       ; GET POINTER TO ERROR MESSAGE
002673  100247              135$:     MOV     R0,(R4)+        ; SAVE
002674  114007                        CLR     R0              ; CLEAR COUNT
002675  104251              140$:     MOV     (R5)+,R1        ; GET UNIT NUMBER
002676                            BMI     145$            ; IF INVALID, BRANCH
002676  070000  002705                ^O070000,145$
002700  115407                        INC     R0              ; INCREMENT COUNT
002701  106207  000004                CMP     #4,R0           ; SEE IF MAX
002703                            BNE     140$            ; IF NOT, LOOP
002703  050000  002675                ^O050000,140$
002705  104671  001627      145$:     MOV     SER18E-1(R0),R1 ; GET POINTER TO CORRECT ERROR MESSAGE
002707  100241                        MOV     R1,(R4)+        ; MOVE INTO OUTPUT BUFFER
002710                            POP     R5              ; RESTORE R5
002710  104265                                                                    MOV (SP)+,R5
002711                            PUSH    R5              ; SAVE R5
002711  100465                                                                    MOV R5,-(SP)
002712  104251              150$:     MOV     (R5)+,R1        ; GET SUBUNIT NUMBER
002713  100241                        MOV     R1,(R4)+        ; SAVE
002714  117407                        DEC     R0              ; DECREMENT COUNT
002715                            BNE     150$            ; IF COUNT INCOMPLETE, BRANCH
002715  050000  002712                ^O050000,150$
002717                            POP     R5              ; RESTORE R5
002717  104265                                                                    MOV (SP)+,R5
002720  105205  000004      155$:     ADD     #4,R5           ; POINT TO NEXT UNIT TABLE
002722  106205  001411                CMP     #UNITS+16.,R5   ; SEE IF ENTIRE TABLE SEARCHED
002724                            BNE     120$            ; IF NOT, BRANCH
002724  050000  002650                ^O050000,120$
002726                            DEVFTL  5000            ; REPORT FATAL ERROR (WILL SHOW UP AS A 5000)
002726  104200  003653  002156                                    MOV     #ER5000,OUT.04
002731  104202  051610                                            MOV     #5000!FTLDEV,R2
002733  104020  002154                                            MOV     R2,OUT.02
002735  104200  002735  002153                                    MOV     #.,OUT.01
002740  104200  060014  002152                                    MOV     #ERRMC,OUT.RQ
002743  104307  002152                MOV     OUT.RQ,R0       ; SET UP FOR REPORT
002745                            CALL    HOSTRQ          ; REPORT ERROR
```

```
            002745  020000  006066                  ^O020000,HOSTRQ
     41     002747  104207  060016                  MOV     #DONE,R0        ; END TEST
     42     002751                                  CALL    HOSTRQ          ; SEND END-OF-TEST TO HOST
            002751  020000  006066                  ^O020000,HOSTRQ
     43     002753  060021                          XFC     EXIT            ; TERMINATE DM
     44
     45     002754  115407                  160$:   INC     R0              ; POINT TO NEXT UNIT TO TEST
     46     002755  104172                          MOV     (R0),R2         ; CHECK NEXT UNIT
     47     002756                                  BPL     90$             ; FIND IN UDA PORT INFORMATION
            002756  030000  002601                  ^O030000,90$
```

```
                                        .SBTTL  TEST 2 START TESTING
                                ;;+
                                ;       FIND OUT IF HOST HAD TO INIT UDA TO GET A PROGRAM FROM AN ATTACHED DRIVE.
                                ;       THE PROGRAM WAS REQUESTED BY THE DRIVE TO BE DOWNLINE LOADED INTO ITS MEMORY.
                                ;       THE INDICATION COMES FROM THE UTOTST RESPONSE.
                                ;
                                ;       A WORD WITH MSB IS SET TO INDICATE THE LAST UNIT ENTRY.  AFTER THAT WORD
                                ;       ANOTHER WORD INDICATES WHEATHER OR NOT A FILE WAS SUPPOSE TO BE DOWN LINE LOADED.
                                ;       A '0' MEANS NORMAL PROCESSING, NO FILE WAS EXPECTED OR REQUESTED.
                                ;       A '1' MEANS A FILE WAS EXPECTED AND IS AVAILABLE.
                                ;       A '2' MEANS A FILE WAS EXPECTED AND IS NOT AVAILABLE.
                                ;
                                ;       R0 IS A POINTER TO THE WORD BEYOND THE LAST TABLE ENTRY
                                ;       (R0) = 100000
                                ;-
002760  104671  000001          T2STRT: MOV     1(R0),R1                ;WAS A FILE EXPECTED?
002762                                  BEQ     PORT0                   ;IF NOT, GO TO NORMAL PROCESSING
002762  010000  003056                  ^O10000,PORT0
002764  115407                          INC     R0                      ;POINT TO INDICATOR IN IN.RQ BUFFER
002765  104075                          MOV     R0,R5                   ;R5 -> INDICATOR
002766  104151                          MOV     (R5),R1                 ;1ST WORD CONTAINS INDICATOR TO SEE IF A FILE WAS EX
002767  106201  000001                  CMP     #1,R1                   ;WAS THE EXPECTED FILE AVAILABLE?
002771                                  BNE     DO.DI7                  ;IF NOT AVAILABLE, GO TO REPORT ERROR
002771  050000  004522                  ^O50000,DO.DI7
                                ; *** FILE WAS THERE, NOW SEE WHERE IN THE TABLE IT WAS SO PORT INDICATOR IS THE SAME?
002773  104207  001371                  MOV     #UNITS,R0               ;R0 -> UNIT TABLE
002775  104651  000001                  MOV     1(R5),R1                ;R1 = UNIT NUMBER
002777  104202  000001                  MOV     #UNIT0,R2               ;R2 = 1ST PORT
003001  114003                          CLR     R3                      ;R3 = UNIT TABLE INDICATOR
003002  114004                  1$:     CLR     R4                      ;R4 KEEPS TRACK OF WHICH SUBUNITS
003003  104270  001624          2$:     MOV     (R0)+,SAVREG            ;STORE UNIT IN SAVE REG
003005                                  BMI     4$                      ;IF NEGATIVE VALUE, DON'T COMPARE
003005  070000  003040                  ^O70000,4$
003007  103300  177400  001624          BIC     HIBYTE,SAVREG           ;CLEAR EXTANEOUS BITS
003012  106301  001624                  CMP     SAVREG,R1               ;DID WE FIND THE DRIVE?
003014                                  BEQ     5$                      ;IF IT IS, WE HAVE FOUND IT
003014  010000  003045                  ^O10000,5$
003016  115404                          INC     R4                      ;ELSE, INCREMENT SUBUNIT POINTER
003017  106204  000004                  CMP     #4,R4                   ;ENDED WITH THIS UNIT ENTRY?
003021                                  BNE     2$                      ;IF NOT, CONTINUE
003021  050000  003003                  ^O50000,2$
003023  105022                  3$:     ADD     R2,R2                   ;ELSE, NEXT PORT SET IN R2
003024  115403                          INC     R3                      ;INCREMENT UNIT TABLE INDICATOR
003025  106203  000004                  CMP     #4,R3                   ;DONE?
003027                                  BNE     1$                      ;IF THIS GETS TO 4, THEN ERROR(UNIT NOT FOUND)
003027  050000  003002                  ^O50000,1$
003031                                  ERRHRD  MS2000,R1
003031  100461                                                                  MOV R1,-(SP)
003032  020000  006261                  ^O20000,RERROR
003034  103720                                                                  .WORD ERHARD+ERRN
003035  010000                                                                  .WORD <PRMS*10000>+MS2000
003036                                  BR      PORT0                   ;START REGULAR TESTING
003036  000000  003056                  ^O0,PORT0
                                ; *** IF HERE, NEGATIVE ENTRY, GO TO NEXT UNIT ENTRY.  ADJUST R0 TO PROPER POINTER
003040  105207  000003          4$:     ADD     #3,R0                   ;R0 -> INTO NEXT UNIT\INCREMENTED BY (R0)+
003042  107047                          SUB     R4,R0                   ;REALIGN TO POINT TO BEGINNING OF POINTER
003043                                  BR      3$                      ;GO BACK INTO THE LOOP
```

```
     003043  000000  003023                  ^O0,3$
47                          ; *** IF HERE, FOUND THE ENTRY, SAVE PERTINENT VALUES, GO DOWN LINE LOAD PROGRAM
48 003045  104650  000001  003623  3$:       MOV     1(R5),LUNIT             ;SAVE LOGICAL UNIT NUMBER
49 003050  104020  001412                    MOV     R2,SDI                  ;SAVE PORT VALUE
50 003052  104030  001411                    MOV     R3,UNITNB               ;SAVE UNIT NUMBER OFFSET IN TABLE
51 003054                                    BR      DO.DI2                  ;GO DOWN LINE LOAD PROGRAM
     003054  000000  004350                  ^O0,DO.DI2
52
```

```
                                        .SBTTL  SEQUENCE THE DIAGNOSTICS TO ALL UNITS SELECTED
                                ;SEQUENCE THE DIAGNOSTICS TO ALL UNITS SELECTED.
                                ;TEST CODE WILL BE CALLED FOR EACH DISK SUBUNIT TO BE TESTED.
                                ;  LUNIT WILL CONTAIN LOGICAL UNIT NUMBER OF DRIVE FOR ERROR REPORTS
                                ;  SDI WILL CONTAIN SDI INTERCONNECT CODE FOR SELECTED DRIVE
                                ;  UNITNB WILL CONTAIN AN EVEN NUMBER FOR TESTING FIRST SUBUNIT OF A DRIVE
                                ;                      AN ODD NUMBER FOR TESTING SECOND SUBUNIT OF A DRIVE

                                ; *** NORMAL PROCESSING
003056                          PORT0:
003056  114001                          CLR     R1              ;START WITH UNIT 0 INDEX

003057  104207  001371          PORT2:  MOV     #UNITS,R0       ; GET POINTER TO UNITS TABLE
003061  105017                          ADD     R1,R0           ; ADD INDEX
003062  104173                          MOV     (R0),R3         ; GET CONTENTS OF TABLE
003063                                  BPL     1$              ; IF THIS UNIT IS PRESENT, BRANCH
003063  030000  003075                  ^O030000,1$
003065  102201  000003                  BIT     #3,R1           ; SEE IF ON SUBUNIT 0 OF UNIT
003067                                  BNE     PORT5           ; IF NOT, TEST NEXT SUBUNIT
003067  050000  003133                  ^O050000,PORT5
003071  105201  000003                  ADD     #3,R1           ; IF NO SUBUNIT 0, THEN NO UNIT - SKIP OTHER SUBUNITS
003073                                  BR      PORT5           ; BYPASS IF NO UNIT
003073  000000  003133                  ^O0,PORT5
003075  102203  040000          1$:     BIT     #40000,R3       ; SEE IF THIS UNIT IS TO BE TESTED
003077                                  BNE     PORT5           ; IF NOT, BRANCH
003077  050000  003133                  ^O050000,PORT5
003101  104010  001411                  MOV     R1,UNITNB       ; STORE UNIT INDEX
003103  104030  001623                  MOV     R3,LUNIT        ; STORE LOGICAL UNIT NUMBER FOR DRIVE
003105  104202  000001                  MOV     #UNIT0,R2       ; GET UNIT 0 INTERCONNECT CODE
003107  110601                          ROR     R1              ; DIVIDE UNITNB BY FOUR
003110  110601                          ROR     R1
003111  103201  177760                  BIC     #LBLONB,R1      ; CLEAR UNUSED BITS
003113  117401                  PORT3:  DEC     R1
003114                                  BMI     PORT4           ; FOR EACH DRIVE OVER 0 SHIFT R2 LEFT
003114  070000  003123                  ^O070000,PORT4
003116  110202                          ROL     R2
003117  103202  000001                  BIC     #1,R2           ; CLEAR CARRY ROTATED INTO REG (IF ANY)
003121                                  BR      PORT3
003121  000000  003113                  ^O0,PORT3
003123  104020  001412          PORT4:  MOV     R2,SDI          ; STORE SDI INTERCONNECT CODE
003125                                  BR      TEST            ; PERFORM TEST ON THIS DRIVE
003125  000000  003146                  ^O0,TEST
                                                                ; TEST RETURNS TO TESTX
003127  104206  002320          TESTX:  MOV     #STACK,SP       ; RESET STACK DO TO JUMPS OUT OF
                                                                ; SUBROUTINES
003131  104301  001411                  MOV     UNITNB,R1       ; GET UNIT INDEX
003133  115401                  PORT5:  INC     R1              ; INCREMENT INDEX
003134  106201  000020                  CMP     #16.,R1         ; CHECK IF 16 DRIVES ALREADY SELECTED
003136                                  BNE     PORT2           ; REPEAT FOR ALL DRIVES
003136  050000  003057                  ^O050000,PORT2
003140  104207  060016          DONECD: MOV     #DONE,R0        ; END OF PROGRAM
003142                                  CALL    HOSTRQ
003142  020000  006066                  ^O020000,HOSTRQ
003144                                  BR      DONECD          ; REPEAT IF RETURNED
003144  000000  003140                  ^O0,DONECD
```

```
                                .SBTTL  INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

                                ;START OF TEST CODE
                                ;
                                ;INPUTS:
                                ;       LUNIT - LOGICAL UNIT NUMBER OF DRIVE UNDER TEST
                                ;       SDI - SDI INTERCONNECT CODE FOR DRIVE

                                ;INITIALIZE THE DRIVE

003146  114003                  TEST:   CLR     R3                      ;R3 IS A DECREMENT COUNTER TO CHECK IF THE
                                                                        ; COMMAND HAS BEEN RECEIVED BY THE DRIVE
003147  104302  001412                  MOV     SDI,R2                  ;GET SDI SELECT CODE
003151  060011                          XFC     DINIT                   ;INITIALIZE THE DRIVE

                                ;WAIT FOR DRIVE TO ASSERT RECEIVER READY
                                ;TIME OUT AFTER ...?

003152  114005                          CLR     R5                      ;GET TIMEOUT COUNTER
003153                          ILOOP:  CALL    RDSTAT                  ;GET DRIVE STATUS
003153  020000  006027                  ^O020000,RDSTAT
003155  102201  010000                  BIT     #10000,R1               ;SEE IF ANY ERRORS
003157                                  BEQ     2$                      ;IF NO ERROR, BRANCH
003157  010000  003202                  ^O010000,2$
003161  102201  004000                  BIT     #4000,R1                ;SEE IF XMIT ERROR
003163  117403                          DEC     R3                      ;ON SEND ERROR, DID WE DEPLETE THE COUNTER?
003164                                  BNE     ILOOP                   ;IF NOT, TRY AGAIN
003164  050000  003153                  ^O050000,ILOOP
003166                                  ERRHRD  MS2001                  ;REPORT INVALID STATUS ERROR
003166  020000  006261                  ^O020000,RERROR
003170  103721                                                                  .WORD ERHARD+ERRN
003171  000043                                                                  .WORD <PRMS*10000>+MS2001
003172                                  BR      DONECD                  ;BRANCH TO DONE
003172  000000  003140                  ^O0,DONECD
003174                          1$:     ERRHRD  MS2002                  ;REPORT XMIT ERROR
003174  020000  006261                  ^O020000,RERROR
003176  103722                                                                  .WORD ERHARD+ERRN
003177  000126                                                                  .WORD <PRMS*10000>+MS2002
003200                                  BR      DONECD                  ;BRANCH TO DONE
003200  000000  003140                  ^O0,DONECD
003202                          2$:
003202  114003                          CLR     R3                      ;R3 IS A DECREMENT COUNTER TO CHECK IF THE
                                                                        ; COMMAND HAS BEEN RECEIVED BY THE DRIVE
003203  102201  000001                  BIT     #RCVRDY,R1              ;CHECK RECEIVER READY LINE
003205                                  BNE     ECHO1                   ;ADVANCE IF ASSERTED
003205  050000  003216                  ^O050000,ECHO1
003207  117405                          DEC     R5                      ;DECREMENT TIME OUT COUNTER
003210                                  BNE     ILOOP                   ;STAY IN LOOP UNTIL SIGNAL SETS OR TIMEOUT
003210  050000  003153                  ^O050000,ILOOP
003212                                  ERRHRD  MS2003                  ;REPORT ERROR
003212  020000  006261                  ^O020000,RERROR
003214  103723                                                                  .WORD ERHARD+ERRN
003215  000165                                                                  .WORD <PRMS*10000>+MS2003
```

ECHO DATA TO DRIVE

```
                                .SBTTL  ECHO DATA TO DRIVE

                                ;ECHO THE FOLLOWING DATA PATTERNS TO THE DRIVE THEN CHECK THE
                                ;RESPONSE FOR THE PROPER DATA RECEIVED:
                                ;       377
                                ;       000
                                ;       252
                                ;       360
                                ;       017

003216  104205  001634        ECHO1:  MOV     #ECHOD,R5               ;GET POINTER TO ECHO DATA
003220  104157                ECHO1A: MOV     (R5),R0                 ;GET ECHO DATA
003221  103207  177400        ECHO2:  BIC     #HIBYTE,R0              ;CLEAR COMMAND FROM WORD
003223  060010                        XFC     ECHO                    ;PERFORM THE ECHO COMMAND

                                ;CHECK FOR TIMEOUT ERROR

003224  115001                        TST     R1                      ;CHECK FOR ERROR
003225                                BEQ     ECHO4                   ;BRANCH IF NONE
003225  010000  003254                ^O10000,ECHO4
003227  102201  000001                BIT     #1,R1                   ;CHECK IF SEND ERROR
003231                                BEQ     ECHO3                   ;BRANCH IF RECEIVE ERROR
003231  010000  003245                ^O10000,ECHO3
003233  117403                        DEC     R3                      ;IF SEND ERROR, IS THE COUNTER BEEN DEPLETED
003234                                BNE     ECHO1A                  ;IF NOT, TRY AGAIN
003234  050000  003220                ^O50000,ECHO1A
003236                                ERRHRD  MS2004,R0               ;REPORT SEND ERROR
003236  100467                                                                MOV R0,-(SP)
003237  020000  006261                ^O20000,RERROR
003241  103724                                                                .WORD ERHARD+ERRN
003242  010235                                                                .WORD <PRMS*10000>+MS2004
003243                                BR      ECHO5
003243  000000  003272                ^O0,ECHO5
003245                        ECHO3:  ERRHRD  MS2005,R0               ;REPORT RECEIVE ERROR
003245  100467                                                                MOV R0,-(SP)
003246  020000  006261                ^O20000,RERROR
003250  103725                                                                .WORD ERHARD+ERRN
003251  010275                                                                .WORD <PRMS*10000>+MS2005
003252                                BR      ECHO5
003252  000000  003272                ^O0,ECHO5

                                ;CHECK DATA RECEIVED

003254  106157                ECHO4:  CMP     (R5),R0                 ;COMPARE DATA RECEIVED WITH
003255                                BEQ     ECHO5                   ; DATA SENT
003255  010000  003272                ^O10000,ECHO5
003257                                ERRHRD  MS2006,(R5),R0          ;REPORT DATA COMPARE ERROR
003257  100467                                                                MOV R0,-(SP)
003260  104010  001624                                                        MOV R1,SAVREG
003262  104151                                                                MOV (R5),R1
003263  100461                                                                MOV R1,-(SP)
003264  104301  001624                                                        MOV SAVREG,R1
003266  020000  006261                ^O20000,RERROR
003270  103726                                                                .WORD ERHARD+ERRN
003271  020340                                                                .WORD <PRMS*10000>+MS2006

                                ;MOVE TO NEXT DATA PATTERN
```

ECHO DATA TO DRIVE

```
      36
      37 003272  114003                ECHO5:  CLR     R3
      38 003273  115405                        INC     R5              ;BUMP TO NEXT DATA TO SEND
      39 003274  104157                        MOV     (R5),R0         ;CHECK IF AT END OF TABLE
      40 003275                                BMI     ECHO2           ;SEND THIS DATA IF NOT
         003275  070000  003221                ^O70000,ECHO2
```

```
                                .SBTTL  GET STATUS COMMAND
                                ;ISSUE GET STATUS COMMAND AND CHECK THAT IT PERFORMS PROPERLY
003277  104200  000002  001413          MOV     #2,SAVSTA               ;COUNTER
003302                                  CALL    GTSTAT                  ;GET STATUS
003302  020000  005242                  ^O020000,GTSTAT
                                ;CLEAR DRIVE ERRORS
003304                          DRCLR1: CALL    CLRDRV                  ; CLEAR DRIVE
003304  020000  005343                  ^O020000,CLRDRV
                                ;ISSUE GET STATUS COMMAND AND CHECK THAT IT PERFORMS PROPERLY
003306                          GSTS3:
003306                                  CALL    GTSTAT                  ;GET STATUS
003306  020000  005242                  ^O020000,GTSTAT
                                ;LOOK AT DATA IN RESPONSE PACKET
003310  104307  001416          GSTS6:  MOV     ST+1,R0                 ;GET TWO WORDS FROM PACKET
003312  104301  001417                  MOV     ST+2,R1
003314  102201  000250                  BIT     #ST.WE+ST.PE+ST.FE,R1   ;CHECK ERROR BITS
003316                                  BEQ     10$                     ;BRANCH IF ALL CLEAR
003316  010000  003332                  ^O010000,10$
003320  117400  001413                  DEC     SAVSTA                  ;TRY ONCE MORE?
003322                                  BNE     DRCLR1                  ;IF OK, TRY AGAIN
003322  050000  003304                  ^O050000,DRCLR1
003324                                  CALL    STOSTA                  ;GO STORE STATUS
003324  020000  005437                  ^O020000,STOSTA
003326                                  ERRHRD  MS2007
003326  020000  006261                  ^O020000,RERROR
003330  103727                                                          .WORD ERHARD+ERRN
003331  000421                                                          .WORD <PRMS*10000>+MS2007
003332  104070  001413          10$:    MOV     R0,SAVSTA               ;SAVE STATUS TO CHECK IF DIAGNOSTIC REQUEST BIT IS S
                                ;ISSUE ONLINE COMMAND AND CHECK THAT IT PERFORMS PROPERLY
003334  104203  002117                  MOV     #CR.ONL,R3              ;R3 -> COMMAND
003336                                  CALL    TALKER                  ;SEND TO DRIVE
003336  020000  006131                  ^O020000,TALKER
003340  115003                          TST     R3                      ;ALL OK?
003341                                  BEQ     30$                     ;IF SO, CHECK RESPONCE CODE
003341  010000  003365                  ^O010000,30$
003343                                  BPL     20$                     ;ELSE, IS MSB SET?/IF NOT, RECEIVE ERROR
003343  030000  003353                  ^O030000,20$
003345                                  ERRHRD  MS2008                  ;REPORT SEND ERROR
003345  020000  006261                  ^O020000,RERROR
003347  103730                                                          .WORD ERHARD+ERRN
003350  000500                                                          .WORD <PRMS*10000>+MS2008
003351                                  BR      GSTS7                   ;
003351  000000  003421                  ^O00,GSTS7
003353                          20$:    CALL    TYPERR
003353  020000  005674                  ^O020000,TYPERR
```

```
     41 003355                                ERRHRD  MS2009,R0,R3          ;REPORT RECEIVE ERROR
        003355  100463                                                                 MOV R3,-(SP)
        003356  100467                                                                 MOV R0,-(SP)
        003357  020000  006261                ^O20000,RERROR
        003361  103731                                                                 .WORD ERHARD+ERRN
        003362  020530                                                                 .WORD <PRMS*10000>+MS2009
     42 003363                                BR      GSTS7                  ;
        003363  000000  003421                ^O00,GSTS7
     43 003365  106207  000176        30$:    CMP     #COMPLT,R0             ;CHECK RESPONSE CODE
     44 003367                                BEQ     GSTS7                  ;IF OK, CONTINUE
        003367  010000  003421                ^O10000,GSTS7
     45 003371  106207  000175                CMP     #UNSSUC,R0             ;IF NOT CORRECT RESPONSE, UNSSUC?
     46 003373                                BNE     35$                    ;IF NOT, UNRECOGNIZED RESPONSE
        003373  050000  003405                ^O50000,35$
     47 003375                                CALL    STOSTA                 ;CHECK STATUS
        003375  020000  005437                ^O20000,STOSTA
     48 003377                                ERRHRD  MS2010                 ;
        003377  020000  006261                ^O20000,RERROR
        003401  103732                                                                 .WORD ERHARD+ERRN
        003402  000564                                                                 .WORD <PRMS*10000>+MS2010
     49 003403                                BR      GSTS7
        003403  000000  003421                ^O00,GSTS7
     50 003405                        35$:    ERRHRD  MS2011,#COMPLT,R0      ;IF NOT, REPORT ERROR
        003405  100467                                                                 MOV R0,-(SP)
        003406  104010  001624                                                         MOV R1,SAVREG
        003410  104201  000176                                                         MOV #COMPLT,R1
        003412  100461                                                                 MOV R1,-(SP)
        003413  104301  001624                                                         MOV SAVREG,R1
        003415  020000  006261                ^O20000,RERROR
        003417  103733                                                                 .WORD ERHARD+ERRN
        003420  020607                                                                 .WORD <PRMS*10000>+MS2011
     51
     52 003421                           GSTS7:
```

```
                                        .SBTTL  GET DRIVE CHARACTERISTICS
                                ;GET DRIVE CHARACTERISTICS
                                
003421  104200  000012  001625            MOV     #10.,SDISTO             ; SET UP TEMPORARY SHORT TIMEOUT VALUE
003424  104203  002111                    MOV     #CR.GCR,R3              ;POINT TO GET CHARS COMMAND
003426                                    CALL    TALKER                  ; INITIATE SDI INTERCHANGE
003426  020000  006131                    ^O20000,TALKER
003430  115003                            TST     R3                      ;SEE IF ERROR OCCURRED
003431                                    BEQ     12$                     ;IF NOT, BRANCH
003431  010000  003455                    ^O10000,12$
003433                                    BPL     11$                     ;IF SO, BRANCH
003433  030000  003443                    ^O30000,11$
003435                                    ERRHRD  MS2012                  ;SEND COMMAND ERROR
003435  020000  006261                    ^O20000,RERROR
003437  103734                                                                    .WORD ERHARD+ERRN
003440  000674                                                                    .WORD <PRMS*10000>+MS2012
003441                                    BR      T00
003441  000000  003511                    ^O0,T00
003443                          11$:      CALL    TYPERR                  ;CHECK WHAT TYPE OF RECEIVE  RROR
003443  020000  005674                    ^O20000,TYPERR
003445                                    ERRHRD  MS2013,R0,R3
003445  100463                                                                    MOV R3,-(SP)
003446  100467                                                                    MOV R0,-(SP)
003447  020000  006261                    ^O20000,RERROR
003451  103735                                                                    .WORD ERHARD+ERRN
003452  020735                                                                    .WORD <PRMS*10000>+MS2013
003453                                    BR      T00
003453  000000  003511                    ^O0,T00
003455  106207  000170            12$:    CMP     #CHRRES,R0              ;CHECK FOR SUCCESSFUL RESPONSE
003457                                    BEQ     T00
003457  010000  003511                    ^O10000,T00
003461  106207  000175                    CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
003463                                    BNE     13$                     ;IF NOT, UNRECOGNIZED RESPONSE
003463  050000  003475                    ^O50000,13$
003465                                    CALL    STOSTA                  ;CHECK STATUS
003465  020000  005437                    ^O20000,STOSTA
003467                                    ERRHRD  MS2014                  ;
003467  020000  006261                    ^O20000,RERROR
003471  103736                                                                    .WORD ERHARD+ERRN
003472  001001                                                                    .WORD <PRMS*10000>+MS2014
003473                                    BR      T00
003473  000000  003511                    ^O0,T00
003475                          13$:      ERRHRD  MS2015,#CHRRES,R0       ;GET CHARACTERISTICS COMMAND FAILED
003475  100467                                                                    MOV R0,-(SP)
003476  104010  001624                                                            MOV R1,SAVREG
003500  104201  000170                                                            MOV #CHRRES,R1
003502  100461                                                                    MOV R1,-(SP)
003503  104301  001624                                                            MOV SAVREG,R1
003505  020000  006261                    ^O20000,RERROR
003507  103737                                                                    .WORD ERHARD+ERRN
003510  021035                                                                    .WORD <PRMS*10000>+MS2015

003511  104307  001415            T00:    MOV     ST+SHRTTO,R0            ; GET SHORT TIMEOUT
003513  103207  177760                    BIC     #LBLONB,R0              ; CLEAR UNUSED BITS
003515                                    CALL    TO                      ; SET UP TIMEOUT
003515  020000  006241                    ^O20000,TO
003517  104070  001625                    MOV     R0,SDISTO               ; SAVE IN SHORT TIMEOUT
```

GET DRIVE CHARACTERISTICS

```
     28 003521  104307  001416                MOV     ST+LONGTO,R0        ; GET LONG TIMEOUT
     29 003523  103207  177760                BIC     #LBLONB,R0          ; CLEAR UNUSED BITS
     30 003525                                CALL    TO                  ; SET UP TIMEOUT
        003525  020000  006241                ^O20000,TO
     31 003527  104070  001626                MOV     R0,SDILTO           ; SAVE IN LONG TIMEOUT
     32
```

```
                                        .SBTTL  CHECK WHICH COMMAND HAS BEEN GIVEN
003531  104307  001413         ST.DIA: MOV     SAVSTA,R0               ;GET STORED STATUS WORD
003533  102207  000040                 BIT     #ST.DR,R0               ;WAS THE DR BIT SET?
003535                                  BNE     DO.DIX                  ;IF SO, GO HANDLE IT WITHIN DIAGNOSE CODE
003535  050000  004257                 ^O50000,DO.DIX
                                ;; *** DO AN INITIAL WRITE MEMORY TO CLEAR THE REGION (ONLY 4 BYTES)
003537  104200  177774  001665         MOV     #FFFC,WRM+1             ;REGION ID
003542  114000  001666                 CLR     WRM+2                   ;OFFSET
003544  104200  000004  001667         MOV     #4,WRM+3                ;BYTE COUNT AND 1ST DATA BYTE
003547  114000  001670                 CLR     WRM+4                   ;DATA
003551  114000  001671                 CLR     WRM+5                   ;DATA
003553  104200  000012  001660         MOV     #10.,CR.WRM+1           ;SET COMMAND BYTE COUNT
003556                                  CALL    WRTMEM                  ;CLEAR BUFFER
003556  020000  004751                 ^O20000,WRTMEM
003560  104200  000007  001660         MOV     #7,CR.WRM+1             ;SET COMMAND BYTE COUNT

                                ; *** NOW GO DIAGNOSE REGION ZERO
003563  114000  002153                 CLR     OUT.01                  ;CLEAR OUT BUFFER TO INITIAL DIAGNOSE COMMANDS
003565  114000  002154                 CLR     OUT.02                  ;
003567  114000  001615                 CLR     DIAG.1                  ;MAKE SURE DIAGNOSE COMMAND RETURNS OP-CODE = 0
                                                                        ; (FOR 1ST TIME THROUGH ONLY)
003571  114000  002217                 CLR     IN.02                   ;CLEAR REGION ID FOR 1ST DIAGNOSE COMMAND
003573                                  BR      DIAGNS                  ;DO DIAGNOSE COMMAND, 1ST THING
003573  000000  003737                 ^O0,DIAGNS

                                ;
                                ; *** RETURN HERE TO GET NEW COMMAND
003575                          GT.CMD:                                 ;
003575  104207  060002                 MOV     #T2CMD,R0               ;SET REQUEST NUMBER
003577                                  CALL    HOSTRQ                  ;ASK HOST FOR PORTS
003577  020000  006066                 ^O20000,HOSTRQ
003601  104307  002216                 MOV     IN.01,R0                ;R0 = RESPONSE CODE;IN.02
                                                                        ; 0 = EXIT ; 1 = WRITE
                                                                        ; 2 = READ ; 3 = DIAGNOSE
003603                                  BEQ     TESTEX                  ; OP = 0 /EXIT
003603  010000  003773                 ^O10000,TESTEX
003605  117407                         DEC     R0                      ; R0 = 1?
003606                                  BEQ     WRITE                   ; IF SO, WRITE
003606  010000  003633                 ^O10000,WRITE
003610  117407                         DEC     R0                      ; R0 = 2?
003611                                  BEQ     READ                    ; IF SO, READ
003611  010000  003701                 ^O10000,READ
003613  117407                         DEC     R0                      ; R0 = 3?
003614                                  BEQ     DIAGNO                  ; IF SO, DIAGNOSE
003614  010000  003765                 ^O10000,DIAGNO
003616                                  ERRHRD  MS2016,IN.01            ; ELSE, ERROR=INVALID INPUT
003616  104010  001624                                                          MOV R1,SAVREG
003620  104301  002216                                                          MOV IN.01,R1
003622  100461                                                                  MOV R1,-(SP)
003623  104301  001624                                                          MOV SAVREG,R1
003625  020000  006261                 ^O20000,RERROR
003627  103740                                                                  .WORD ERHARD+ERRN
003630  011133                                                                  .WORD <PRMS*10000>+MS2016
003631                                  BR      GT.CMD
003631  000C00  003575                 ^O0,GT.CMD
```

```
                                                .SBTTL  MEMORY WRITE
                                        ;;+
                                        ;
                                        ;       INPUT   IN.02 = REGION ID
                                        ;               IN.03 = OFFSET
                                        ;               IN.04 = DATA BYTE
                                        ;
                                        ;       OUTPUT  OUT.RQ HAS DAT SET FOR T2CMD
                                        ;;-
                                        WRITE:
                                                MOV     IN.02,WRM+1             ;REGION ID
                                                MOV     IN.03,WRM+2             ;OFFSET
                                                MOV     IN.04,R0                ;R0 = BYTE OF DATA IN LO BYTE
                                                SWAB    R0                      ;R0 IS IN HI BYTE
                                                BIC     #LOBYTE,R0              ;CLEAR LO BYTE
                                                BIS     #1,R0                   ;SET BYTE COUNT
                                                MOV     R0,WRM+3                ;SET WORD IN PACKET
                                                CLR     WRM+4
                                                CLR     WRM+5
                                                MOV     #7,CR.WRM+1             ;SET COMMAND BYTE COUNT
                                        ;
                                                CALL    WRTMEM
                                                ^O20000,WRTMEM
                                                MOV     #1,OUT.02               ;R0 = OP CODE
                                                MOV     LUNIT,OUT.01            ;DRIVE NUMBER
                                                CLR     OUT.03                  ;
                                                CLR     OUT.04                  ;
                                                CLR     OUT.05                  ;
                                                BR      GT.CMD                  ;GO SEND REQUEST
                                                ^O0,GT.CMD
```

```
                                                .SBTTL  MEMORY READ
                                        ;;+
                                        ;
                                        ;       INPUT   IN.02 HAS REGION ID
                                        ;               IN.03 HAS OFFSET
                                        ;
                                        ;       OFFSET  SETS UP OUTBUFFER FOR T2CMD
                                        ;;-
003701                                  READ:
003701  104300  002217  002101                 MOV     IN.02,RDM+1             ;REGION ID
003704  104300  002220  002102                 MOV     IN.03,RDM+2             ;OFFSET
003707  104200  000001  002103                 MOV     #1,RDM+3                ;BYTE COUNT
                                        ;
003712                                         CALL    RDMEM
003712  020000  004642                         ^O20000,RDMEM
                                        ; *** SET UP RESPONSE PACKET
003714  104300  001623  002153                 MOV     LUNIT,OUT.01            ;SET UP REQUEST
003717  104200  000002  002154                 MOV     #2,OUT.02               ;RETURN OPCODE SET IN BUFFER
003722  104307  001415                         MOV     ST,R0                   ;R0 = BYTE COUNT + DATA
003724  110707                                 SWAB    R0                      ;DATA IN LO BYTE
003725  103207  177400                         BIC     #HIBYTE,R0              ;CLEAR BYTE COUNT (1 BYTE ONLY SENT)
003727  104070  002155                         MOV     R0,OUT.03               ;STORE IN BUFFER FOR HOST
003731  114000  002156                         CLR     OUT.04                  ;
003733  114000  002157                         CLR     OUT.05                  ;
003735                                         BR      GT.CMD                  ;GO SEND REQUEST
003735  000000  003575                         ^O0,GT.CMD
```

SEND DIAGNOSE COMMAND

```
      1
      2 003737  102200  000001  001413  DIAGNS: BIT     #ST.RU,SAVSTA           ;CHECK IF DRIVE IS SPINNABLE (RUN SWITCH IN?)
      3 003742                                  BEQ     3$                      ;ONLY DO DIAGNOSE ONCE IF NOT SPINNABLE
        003742  010000  003765                  ^O10000,3$
      4                                 ; *** DRIVE IS SPINNABLE
      5 003744  104302  001412                  MOV     SDI,R2                  ;R2 HAS CURRENT SDI PORT INDEX
      6 003746  102200  000020  001413          BIT     #ST.SR,SAVSTA           ;HAS THE DRIVE BEEN SPUN UP?
      7 003751                                  BNE     2$                      ;IF NOT ALREADY SPUN UP, DO DIAGNOSE TWICE
        003751  050000  003763                  ^O50000,2$
      8                                 ; *** DRIVE HAS NOT BEEN SPUN UP
      9 003753                          1$:     CALL    DIAGDR                  ;DO A DIAGNOSE COMMAND
        003753  020000  003775                  ^O20000,DIAGDR
     10 003755  104302  001412                  MOV     SDI,R2                  ;RESTORE PORT INDEX
     11 003757                                  CALL    RUNDR                   ;SPIN UP THE DRIVE
        003757  020000  005052                  ^O20000,RUNDR
     12 003761                                  BR      3$
        003761  000000  003765                  ^O0,3$
     13                                 ; *** DRIVE HAS BEEN SPUN UP
     14 003763                          2$:     CALL    RECAL                   ;ELSE DO A RECAL
        003763  020000  005146                  ^O20000,RECAL
     15 003765                          3$:
     16 003765  104302  001412          DIAGNO: MOV     SDI,R2                  ;RESTORE PORT INDEX
     17 003767                                  CALL    DIAGDR                  ;DO A DIAGNOSE COMMAND
        003767  020000  003775                  ^O20000,DIAGDR
     18 003771                                  BR      GT.CMD                  ;GET COMMAND
        003771  000000  003575                  ^O0,GT.CMD
     19
     20                                 ;END OF TESTING THIS DRIVE
     21
     22 003773                          TESTEX: BR TESTX                        ;GO BACK TO DRIVE SEQUENCER
        003773  000000  003127                  ^O0,TESTX
     23
```

```
                                        .SBTTL  TEST 2 SPECIFIC ROUTINES
                                        .SBTTL  DIAGNOSE COMMAND PROCESSING
                                ;:+
                                ;
                                ;       INPUT   IN.02 HAS REGION ID
                                ;
                                ;       OUTPUT  OUT.RQ IS SET UP FOR T2CMD
                                ;               IF ERROR, R2 = 1
                                ;                   ELSE, R2 = 0
                                ;:-
003775                          DIAGDR:
003775  104200  177777  001622          MOV     #177777,SENDHR          ;SET SENDHR NONZERO FOR DIAGNOSE COMMAND
004000  104300  002217  002134          MOV     IN.02,DIA+1             ;SET UP MEMORY REGION ID
004003  104203  002126                  MOV     #CR.DIA,R3              ;R3->DIAGNOSE PACKET
004005  104302  001412                  MOV     SDI,R2                  ;R2 = PORT
004007                                  CALL    TALKER                  ;SEND COMMAND AND RECEIVE RESPONSE FROM DRIVE
004007  020000  006131                  ^O020000,TALKER
004011  114000  001622                  CLR     SENDHR
004013  115003                          TST     R3                      ;SEE IF ERROR OCCURRED
004014                                  BEQ     2$                      ;IF NO ERRORS, BRANCH
004014  010000  004041                  ^O010000,2$
004016  110203                          ROL     R3                      ;SHIFT HIGH BIT INTO CARRY BIT
004017                                  BCC     1$                      ;IF CARRY CLEAR (RECEIVE ERROR) BRANCH
004017  040000  004027                  ^O040000,1$
                                ; *** REPORT TRANSMISSION ERROR
004021                                  ERRHRD  MS2017
004021  020000  006261                  ^O020000,RERROR
004023  103741                                                          .WORD ERHARD+ERRN
004024  001224                                                          .WORD <PRMS*10000>+MS2017
004025                                  BR      4$
004025  000000  004101                  ^O0,4$
                                ; *** REPORT RECEPTION ERROR
004027                          1$:     CALL    TYPERR
004027  020000  005674                  ^O020000,TYPERR
004031                                  ERRHRD  MS2018,R0,R3
004031  100463                                                          MOV R3,-(SP)
004032  100467                                                          MOV R0,-(SP)
004033  020000  006261                  ^O020000,RERROR
004035  103742                                                          .WORD ERHARD+ERRN
004036  021255                                                          .WORD <PRMS*10000>+MS2018
004037                                  BR      4$
004037  000000  004101                  ^O0,4$
                                ; *** CHECK RESPONSE OP-CODE FROM DRIVE
004041                          2$:
004041  106207  000374                  CMP     #DIA.EN,R0              ;CHECK RESPONSE CODE
004043                                  BEQ     5$                      ;IF EQUAL, BRANCH
004043  010000  004107                  ^O010000,5$
004045  106207  000176                  CMP     #COMPLT,R0              ; WAS IT COMPLETED?
004047                                  BEQ     5$                      ; IF SO, BRANCH
004047  010000  004107                  ^O010000,5$
004051  106207  000175                  CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
004053                                  BNE     3$                      ;IF NOT, UNRECOGNIZED RESPONSE
004053  050000  004065                  ^O050000,3$
004055                                  CALL    STOSTA                  ;CHECK STATUS
004055  020000  005437                  ^O020000,STOSTA
004057                                  ERRHRD  MS2019                  ;
004057  020000  006261                  ^O020000,RERROR
```

```
                004061  103743                                                  .WORD ERHARD+ERRN
                004062  001312                                                  .WORD <PRMS*10000>+MS2019
            39  004063                          BR      4$
                004063  000000  004101          ^O0,4$
            40  004065                  3$:     ERRHRD  MS2020,#DIA.EN,R0
                004065  100467                                                  MOV R0,-(SP)
                004066  104010  001624                                          MOV R1,SAVREG
                004070  104201  000374                                          MOV #DIA.EN,R1
                004072  100461                                                  MOV R1,-(SP)
                004073  104301  001624                                          MOV SAVREG,R1
                004075  020000  006261          ^O020000,RERROR
                004077  103744                                                  .WORD ERHARD+ERRN
                004100  021336                                                  .WORD <PRMS*10000>+MS2020
            41  004101                  4$:     CALL    CLRDRV
                004101  020000  005343          ^O020000,CLRDRV
            42  004103  104202  000001          MOV     #1,R2
            43  004105                          RETURN
                004105  000000  000000          ^O0,0
```

```
                                        .SBTTL  DIAGNOSE/READ MEMORY TO SEE IF ERROR OCCURRED
                                ; *** DO A READ MEMORY SDI COMMAND
004107  114000  001622          5$:     CLR     SENDHR
004111  104300  001415  002101          MOV     ST,RDM+1                ;REGION ID SET
004114  114000  002102                  CLR     RDM+2                   ;CLEAR OFFSET
004116  104200  000010  002103          MOV     #8.,RDM+3               ;BYTE COUNT = 8
004121                                  CALL    RDMEM                   ;READ THE MEMORY
004121  020000  004642                  ^O20000,RDMEM
004123  115002                          TST     R2                      ;ALL OK?
004124                                  BNE     DO.DC4                  ;IF NOT, DO DRIVE CLEAR/TRY NEXT DRIVE
004124  050000  004216                  ^O50000,DO.DC4
004126  104302  001412                  MOV     SDI,R2                  ;RESTORE PORT INDICATOR
004130  104204  002157                  MOV     #OUT.05,R4              ;R4 -> OUT BUFFER
004132                                  CALL    CONMEM                  ;CONVERT MEMORY
004132  020000  005472                  ^O20000,CONMEM
                                ; *** DATA NOW IN OUT.RQ
004134  104303  002160                  MOV     OUT.06,R3               ;R3 HAD ET & DA FLAGS
                                ; *** CHECK IF DATA IS AVAILABLE FOR INFORMATION
004136  102203  040000                  BIT     #DATAVL,R3              ;IS DATA AVAILABLE?
004140                                  BEQ     6$                      ;IF NOT, CONTINUE WITH THIS DRIVE
004140  010000  004162                  ^O10000,6$
004142  104200  000010  002102          MOV     #8.,RDM+2               ;PREVIOUS BYTE COUNT
004145                                  CALL    GETBCN                  ;GET NEW BYTE COUNT
004145  020000  005536                  ^O20000,GETBCN
004147                                  CALL    RDMEM                   ;READ MEMORY XFC
004147  020000  004642                  ^O20000,RDMEM
004151  115002                          TST     R2                      ;ALL OK?
004152                                  BNE     DO.DC4                  ;IF NOT, DO DRIVE CLEAR/ELSE, DROP DRIVE
004152  050000  004216                  ^O50000,DO.DC4
004154  104302  001412                  MOV     SDI,R2                  ;RESTORE PORT INDICATOR
004156  104204  002163                  MOV     #OUT.09,R4              ;R4 -> OUT BUFFER
004160                                  CALL    CONMEM                  ;CONVERT MEMORY
004160  020000  005472                  ^O20000,CONMEM
004162  102203  100000          6$:     BIT     #ERRTYP,R3              ;WAS ET SET? WITH NO ERROR NUMBER???
004166                                  BEQ     7$                      ;IF NOT, CONTINUE
004166  010000  004174                  ^O10000,7$
                                ; *** REPORT AN ERROR
004166                                  ERRHRD  MS2021
004166  020000  006261                  ^O20000,RERROR
004170  103745                                                          .WORD ERHARD+ERRN
004171  001424                                                          .WORD <PRMS*10000>+MS2021
004172                                  BR      DO.DC4                  ;GO DO DRIVE CLEAR
004172  000000  004216                  ^O00,DO.DC4
004174  102203  040000          7$:     BIT     #DATAVL,R3              ;IS DATA AVAILABLE?
004176                                  BEQ     DO.DC4                  ;IF NOT, DO DRIVE CLEAR
004176  010000  004216                  ^O10000,DO.DC4
004200                                  MSSGE   MSG1
004200  104300  001623  002153                  MOV     LUNIT,OUT.01
004203  104200  003242  002154                  MOV     #MSG1,OUT.02
004206  100467                                          MOV R0,-(SP)
004207  100461                                          MOV R1,-(SP)
004210  104207  060015                          MOV     #MESSAG,R0
004212  020000  006066                  ^O20000,HOSTRQ
004214  104261                                          MOV (SP)+,R1
004215  104267                                          MOV (SP)+,R0
```

```
                                        .SBTTL  DIAGNOSE/DO A DRIVE CLEAR
                                ; *** DO DRIVE CLEAR, GET STATUS AND CHECK IF DR BIT IS SET
004216  104200  000004  001627  DO.DC4: MOV     #4,SAFWRD               ;SAFWRD = # MAXIMUM DRIVE CLEAR TRIES
004221  104302  001412          DO.DCL: MOV     SDI,R2                  ;RESTORE PORT INDICATOR
004223                                  CALL    CLRDRV                  ;CALL DRIVE CLEAR
004223  020000  005343                  ^O020000,CLRDRV
004225  115003                          TST     R3                      ;ERROR?
004226                                  BEQ     1$                      ;IF NOT, CONTINUE
004226  010000  004236                  ^O010000,1$
004230  117400  001627                  DEC     SAFWRD                  ;REACHED MAXIMUM # OF DRV CLR TRIES?
004232                                  BNE     DO.DCL                  ;IF NOT, TRY AGAIN
004232  050000  004221                  ^O050000,DO.DCL
004234                                  BR      DO.DIO                  ; ELSE, ERROR
004234  000000  004607                  ^O00,DO.DIO
004236                          1$:     CALL    GTSTAT                  ;CALL GET STATUS
004236  020000  005242                  ^O020000,GTSTAT
004240  115003                          TST     R3                      ;ERROR?
004241                                  BEQ     2$                      ;IF NOT, CONTINUE
004241  010000  004251                  ^O010000,2$
004243  117400  001627                  DEC     SAFWRD                  ;REACHED MAXIMUM # OF DRV CLR TRIES?
004245                                  BNE     DO.DCL                  ;IF NOT, TRY AGAIN
004245  050000  004221                  ^O050000,DO.DCL
004247                                  BR      DO.DIO                  ; ELSE, ERROR
004247  000000  004607                  ^O00,DO.DIO
004251  104307  001416          2$:     MOV     ST+1,R0                 ;GET STATUS WORD
004253  102207  000040                  BIT     #ST.DR,R0               ;DR SET?
004255                                  BEQ     DO.DIO                  ;IF NOT, EXIT
004255  010000  004607                  ^O010000,DO.DIO
                                ; *** IF DR BIT IS SET, READ 6 BYTES FROM REGION ID FFFD
004257  104200  177775  002101  DO.DIX: MOV     #FFFD,RDM+1             ;FFFD (REGION ID) STORED IN READ MEM PACKET
004262  114000  002102                  CLR     RDM+2                   ;OFFSET = 0
004264  104200  000006  002103          MOV     #6,RDM+3                ;BYTE COUNT = 6
004267                                  CALL    RDMEM                   ;READ MEMORY
004267  020000  004642                  ^O020000,RDMEM
004271  115002                          TST     R2                      ;WAS THERE AN ERROR?
004272                                  BEQ     3$                      ;IF THERE WAS NOT, CONTINUE
004272  010000  004276                  ^O010000,3$
004274                                  BR      DO.DIO                  ;DO A DRIVE CLEAR
004274  000000  004607                  ^O00,DO.DIO
004276  104302  001412          3$:     MOV     SDI,R2                  ;RESTORE PORT INDICATOR
004300  104204  001415                  MOV     #ST,R4                  ;R4 -> ST
004302                                  CALL    CONMEM                  ;CONVERT MEMORY
004302  020000  005472                  ^O020000,CONMEM
                                ; *** GET REGION ID AND PROGRAM NAME IF ANY
004304  104300  001415  001414          MOV     ST,SAVRID               ;SAVE REGION ID
004307  115000  001416                  TST     ST+1                    ;IS CHARACTER ENCODED?
004311                                  BNE     DO.DI1                  ;IF SO, GO GET FILE FROM HOST
004311  050000  004317                  ^O050000,DO.DI1
004313  115000  001417                  TST     ST+2                    ;IF 1ST WORD 0, IS THE SECOND?
004315                                  BEQ     DO.DI3                  ;IF SO, GO DIAGNOSE REGION
004315  010000  004515                  ^O010000,DO.DI3
```

```
                                                .SBTTL  DIAGNOSE/GET PROGRAM NAME SPECIFIED BY DRIVE AND DOWNLINE LOAD
                                        ; *** GET THE PROGRAM WHICH NAME WAS SPECIFIED BY THE DRIVE
004317                                  DO.DI1:
004317  104300  001414  001665                MOV     SAVRID,WRM+1            ;STORE REGION ID IN THE WRITE PACKET
004322  104203  002153                        MOV     #OUT.01,R3              ;R3 -> INPUT DATA
004324  104307  001623                        MOV     LUNIT,R0
004326  100237                              1 MOV     R0,(R3)+                ;SET UNIT NUMBER
004327  114007                              2 CLR     R0
004330  100237                              3 MOV     R0,(R3)+                ;CLEAR VALUE
004331  104307  001415                      4 MOV     ST,R0
004333  100237                              5 MOV     R0,(R3)+                ;SAVE REGION ID
004334  104307  001416                      6 MOV     ST+1,R0
004336  100237                              7 MOV     R0,(R3)+                ;1ST HALF NAME SAVED
004337  104307  001417                      8 MOV     ST+2,R0
004341  100237                              9 MOV     R0,(R3)+                ;2ND HALF NAME SAVED
004342  104207  060001                     10 MOV     #T2DLL,R0               ;SET R0 = TEST 2 DOWNLINE LOAD
004344                                     11 CALL    HOSTRQ
004344  020000  006066                     12 ^O20000,HOSTRQ
                                        ; *** IF HERE, DIDN'T REINIT UDA TO GET DATA
004346  104205  002216                     13 MOV     #IN.01,R5               ;R5 -> INPUT BUFFER
                                        ; *** DOWN LINE LOAD PROGRAM
004350  114000  001666                  DO.DI2: CLR   WRM+2                   ;CLEAR OFFSET
004352  104157                             14 MOV     (R5),R0                 ;IS THE PROGRAM THERE?
004353  117407                             15 DEC     R0
004354                                     16 BNE     DO.DI8                  ;IF NOT, REPORT ERROR
004354  050000  004525                     17 ^O50000,DO.DI8
                                        ; *** GET THE PROGRAM FROM THE HOST
004356  104657  000004                  1$:    MOV     4(R5),R0                ;UNIBUS ADDRESS PA
004360  104651  000005                     18 MOV     5(R5),R1                ;UNIBUS ADDRESS EA
004362  104652  000006                     19 MOV     6(R5),R2                ;BYTE COUNT
004364                                     20 BNE     2$                      ;IF BYTE COUNT IS NOT ZERO, CONTINUE
004364  050000  004372                     21 ^O50000,2$
004366                                     22 ERRHRD  MS2022                  ;ELSE, REPORT THE ERROR
004366  020000  006261                     23 ^O20000,RERROR
004370  103746                                                                     .WORD ERHARD+ERRN
004371  001511                                                                     .WORD <.PRMS*10000>+MS2022
004372  104203  001415                  2$:    MOV     #ST,R3                  ;POINTER TO INPUT BUFFER
004374  106202  000200                     24 CMP     #STSIZE,R2              ;R2 > MAX BUFFER SIZE?
004376                                     25 BPL     3$                      ;IF NOT, CONTINUE
004376  030000  004402                     26 ^O30000,3$
004400  104202  000200                     27 MOV     #STSIZE,R2              ;SET MAX BUFFER SIZE IN R2
004402  060013                          3$:    XFC     UREAD                   ;DO A UNIBUS READ
                                        ; *** SET UP SDI COMMAND PACKETS
004403  104020  001660                     28 MOV     R2,CR.WRM+1             ;SET BYTE COUNT IN WRM PACKET
004405  105200  000006  001660            29 ADD     #6,CR.WRM+1             ;ADD TO COUNT TO INCLUDE PACKET LENGTH(1),
                                                                               ; SDI OPCODE(1), REGION ID(2), AND OFFSET(2)
004410  104650  000003  001665            30 MOV     3(R5),WRM+1             ;SET REGION ID
004413  104020  001667                     31 MOV     R2,WRM+3                ;SET BYTE COUNT IN BUFFER
004415  104303  001415                     32 MOV     ST,R3                   ;1ST DATA WORD IN R3
004417  110703                             33 SWAB    R3                      ;1ST DATA WORD IN UPPER BYTE
004420  103203  000377                     34 BIC     #LOBYTE,R3              ;CLEAR LOW BYTE
004422  101030  001667                     35 BIS     R3,WRM+3                ;SET DATA IN 1ST BUFFER WORD OF WRT MEM PACKET
004424  104020  001624                     36 MOV     R2,SAVREG               ;SAVE BYTE COUNT
004426  117402                             37 DEC     R2                      ;ADJUST BYTE COUNT
004427  103200  000377  001415            38 BIC     #LOBYTE,ST              ;CLEAR LOW BYTE OF 1ST DATA WORD OF PROGRAM
004432  101020  001415                     39 BIS     R2,ST                   ;REPLACE IT BY THE BYTE COUNT OF THE REST
004434  104204  001670                     40 MOV     #WRM+4,R4               ;R4 -> OUT BUFFER
```

```
     51 004436                                          CALL    CONMEM          ;CONVERT MEMORY
        004436  020000  005472                          ^O20000,CONMEM
     52                                         ; *** SEND TO THE DRIVE
     53 004440  104302  001412                          MOV     SDI,R2          ;RESTORE PORT INDICATOR
     54 004442                                          CALL    WRTMEM          ;SEND TO DRIVE
        004442  020000  004751                          ^O20000,WRTMEM
     55 004444  115002                                  TST     R2              ;ALL OK?
     56 004445                                          BNE     DO.DC4          ;IF NOT, GO CLEAR DRIVE
        004445  050000  004216                          ^O50000,DO.DC4
     57 004447  104302  001624                          MOV     SAVPEG,R2       ;R2 = BYTE COUNT
     58 004451  105020  001666                          ADD     R2,WRM+2        ;ADJUST OFFSET
     59 004453  104657  000006                          MOV     6(R5),R0
     60 004455  107027                                  SUB     R2,R0
     61 004456  100657  000006                          MOV     R0,6(R5)        ;DECREMENT TOTAL BYTE COUNT/DONE?
     62 004460                                          BEQ     5$              ;IF 0, EXIT
        004460  010000  004503                          ^O10000,5$
     63 004462                                          BMI     5$              ;IF NEG, EXIT
        004462  070000  004503                          ^O70000,5$
     64 004464  104657  000004                          MOV     4(R5),R0
     65 004466  105027                                  ADD     R2,R0
     66 004467  100657  000004                          MOV     R0,4(R5)        ;ELSE, ADJUST UNIBUS ADDRESS TO READ FROM
     67 004471  106027                                  CMP     R2,R0           ;IS PA ADDRESS LESS THAN BUFFER SIZE?
     68 004472                                          BMI     1$              ;IF IT IS NOT, CONTINUE
        004472  070000  004356                          ^O70000,1$
     69 004474  104657  000005                          MOV     5(R5),R0
     70 004476  115407                                  INC     R0              ;ELSE, INCREMENT EA ADDRESS (CROSSED 0 BOUNDARY)
     71 004477  100657  000005                          MOV     R0,5(R5)
     72 004501                                  4$:     BR      1$              ;READ IN NEXT BUFFER
        004501  000000  004356                          ^O00,1$
     73
     74                                         ; *** SET UP DIAGNOSE COMMAND
     75 004503  114000  001666                  5$:     CLR     WRM+2                   ;REINIT OFFSET
     76 004505  104200  000007  001660                  MOV     #7,CR.WRM+1             ;REINIT BYTE COUNT OF INSTRUCTION
     77 004510  104650  000003  002217                  MOV     3(R5),IN.02             ;SET UP REGION ID
     78 004513                                          BR      DIAGDR                  ;SEND DIAGNOSE COMMAND
        004513  000000  003775                          ^O00,DIAGDR
     79
     80                                         ; *** JUST RECEIVED THE REGION ID TO DIAGNOSE TO.  GO DIAGNOSE IT.
     81 004515  104300  001414  002217  DO.DI3: MOV     SAVRID,IN.02            ;STORE REGION ID FOR DIAGNOSE COMMAND
     82 004520                                          BR      DIAGDR          ;GO DIAGNOSE
        004520  000000  003775                          ^O00,DIAGDR
```

```
                                        .SBTTL  DIAGNOSE/REPORT ERROR -- NO DOWNLINE LOAD PROGRAM
                                ; *** NO DOWN LINE LOAD PROGRAM AFTER REINITED UDA
004522  104650  000001  001411  DO.DI7: MOV     1(R5),UNITNB            ;SET UP LOGICAL UNIT NUMBER

                                ; *** NO DOWN LINE LOAD PROGRAM, REPORT ERROR
004525  104654  000007          DO.DI8: MOV     7(R5),R4                ;R4 = 1ST WORD OF PROGRAM NAME
004527                                  CALL    DIV50                   ;DIVIDE BY 50 AND GET 3RD CHARACTER
004527  020000  005747                  ^O20000,DIV50
004531  110707                          SWAB    R0                      ;PUT IN PROPER BYTE
004532  104070  001617                  MOV     R0,NAM.2                ;SET 3RD CHARACTER
004534                                  CALL    DIV50                   ;GET 2ND CHARACTER
004534  020000  005747                  ^O20000,DIV50
004536  104070  001620                  MOV     R0,NAM.3                ;SET 2ND CHARACTER
004540  104047                          MOV     R4,R0                   ;R0 = RAD50 OF 1ST CHARACTER
004541                                  CALL    FNDASC                  ;GET ASCII EQUIVALENT
004541  020000  005764                  ^O20000,FNDASC
004543  110707                          SWAB    R0                      ;PUT IN PROPER BYTE
004544  101070  001620                  BIS     R0,NAM.3                ;SET 1ST CHARACTER
004546  104654  000010                  MOV     10(R5),R4               ;R4 = 2ND WORD OF PROGRAM NAME
004550                                  CALL    DIV50                   ;GET 6TH CHARACTER
004550  020000  005747                  ^O20000,DIV50
004552  104070  001616                  MOV     R0,NAM.1                ;SET 6TH CHARACTER
004554                                  CALL    DIV50                   ;GET 5TH CHARACTER
004554  020000  005747                  ^O20000,DIV50
004556  110707                          SWAB    R0                      ;PUT IN PROPER BYTE
004557  101070  001616                  BIS     R0,NAM.1                ;SET 5TH CHARACTER
004561  104047                          MOV     R4,R0                   ;R0 = RAD50 OF 4TH CHARACTER
004562                                  CALL    FNDASC                  ;FIND ASCII EQUIVALENT
004562  020000  005764                  ^O20000,FNDASC
004564  101070  001617                  BIS     R0,NAM.2                ;SET 4TH CHARACTER
004566                                  ERRHRD  MS2023,NAM.1,NAM.2,NAM.3 ;REPORT ERROR
004566  104010  001624                                                          MOV R1,SAVREG
004570  104301  001620                                                          MOV NAM.3,R1
004572  100461                                                                  MOV R1,-(SP)
004573  104301  001617                                                          MOV NAM.2,R1
004575  100461                                                                  MOV R1,-(SP)
004576  104301  0016:6                                                         MOV NAM.1,R1
004600  100461                                                                  MOV R1,-(SP)
004601  104301  001624                                                          MOV SAVREG,R1
004603  020000  006261          ^O20000,RERROR
004605  103747                                                                  .WORD ERHARD+ERRN
004606  031554                                                                  .WORD <PRMS*10000>+MS2023
```

```
                                        .SBTTL  DIAGNOSE/SET UP RESPONSE TO HOST AND EXIT
                                ; *** SET UP RESPONSE PACKET
004607  104302  001412         DO.DIO: MOV     SDI,R2                  ;RESTORE PORT INDICATOR
004611  104300  001623  002153         MOV     LUNIT,OUT.01            ;SAVE UNIT NUMBER
004614  115000  001615                 TST     DIAG.1                  ;IF DIAG.1 = 0?
004616                                 BNE     1$                      ;IF NOT, SKIP
004616  050000  004624                 ^O50000,1$
004620  114000  002154                 CLR     OUT.02                  ;SET UP PROPER OP-CODE VALUE
004622                                 BR      2$                      ;CONTINUE
004622  000000  004627                 ^O00,2$
004624  104200  000003  002154 1$:     MOV     #3,OUT.02               ;SET UP REQUEST
004627                         2$:
004627  104300  001415  002155         MOV     ST,OUT.03               ;REGION ID SET
004632  114000  002156                 CLR     OUT.04                  ;
004634  104200  177777  001615         MOV     #177777,DIAG.1          ;MAKE SURE DIAG.1 HAS NONE ZERO VALUE IN IT
                                                                       ; FOR NEXT TIME THROUGH
004637  114002                         CLR     R2
004640                                 RETURN
004640  000000  000000                 ^O00,0
```

```
                                        .SBTTL  READ MEMORY SUBROUTINE
                                ;:+
                                ;       ROUTINE NAME:   RDMEM
                                ;
                                ;       DESCRIPTION:
                                ;       THIS ROUTINE DOES THE XFC READ MEMORY (FROM A SPECIFIED REGION
                                ;       AND OFFSET) OF DATA FROM THE DRIVE.  THIS ROUTINE CALLS THE
                                ;       SEND AND RCV (TALKER) ROUTINE AND JUDGES IF THE SDI COMMAND
                                ;       WAS PROPERLY SENT.
                                ;
                                ;       INPUT:  ALL PARAMETERS FOR THE READ MEMORY COMMAND
                                ;               MUST BE SET.  THAT MEANS REGION ID, THE OFFSET,
                                ;               AND THE BYTE COUNT MUST BE GIVEN THE APPRORIATE VALUES
                                ;               AND STORED IN THE RDM PACKET
                                ;
                                ;       OUTPUT: R2 = 0 MEANS ALL IS OK
                                ;               R2 NOT = 0, READ FAILED (EITHER SEND OR RECIEVE OF THE COMMAND)
                                ;               ST HAS THE READ DATA
                                ;:-
004642                          RDMEM:
004642                                  PUSH    <R0,R1,R3>              ;SAVE THESE REGISTERS
004642  100467                                                                  MOV R0,-(SP)
004643  100461                                                                  MOV R1,-(SP)
004644  100463                                                                  MOV R3,-(SP)
004645  104203  002073                 MOV     #CR.RDM,R3              ;R3 -> RDM PACKET
004647  104302  001412                 MOV     SDI,R2                  ;R2 = PORT
004651                                  CALL    TALKER                  ;SEND COMMAND AND RECEIVE RESPONSE
004651  020000  006131                 ^O020000,TALKER
004653  115003                          TST     R3                      ;ERRORS?
004654                                  BEQ     2$                      ;IF NONE SO FAR, CONTINUE
004654  010000  004701                 ^O010000,2$
004656  110203                          ROL     R3                      ;ERROR ON SEND?
004657                                  BCC     1$                      ;IF CARRY CLEAR (RECEIVE ERROR) BRANCH
004657  040000  004667                 ^O040000,1$
                                ; *** REPORT TRANSMISSION ERROR
004661                                  ERRHRD  MS2024
004661  020000  006261                 ^O020000,RERROR
004663  103750                                                                  .WORD ERHARD+ERRN
004664  001621                                                                  .WORD <PRMS*10000>+MS2024
004665                                  BR      4$                      ;ERROR EXIT
004665  000000  004742                 ^O0,4$
                                ; *** REPORT RECEPTION ERROR
004667                          1$:     CALL    TYPERR                  ;GO CHECK TYPE OF ERROR
004667  020000  005674                 ^O020000,TYPERR
004671                                  ERRHRD  MS2025,R0,R3
004671  100463                                                                  MOV R3,-(SP)
004672  100467                                                                  MOV R0,-(SP)
004673  020000  006261                 ^O020000,RERROR
004675  103751                                                                  .WORD ERHARD+ERRN
004676  021653                                                                  .WORD <PRMS*10000>+MS2025
004677                                  BR      4$                      ;ERROR EXIT
004677  000000  004742                 ^O0,4$
                                ; *** CHECK RESPONSE OP-CODE FROM DRIVE
004701  106207  000162            2$:     CMP     #RDM.EN,R0              ;CHECK RESPONSE
004703                                  BEQ     3$                      ;IF IT MATCHES, OK, EXIT
004703  010000  004737                 ^O010000,3$
004705  106207  000175                 CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
```

```
 40 004707                              BNE     6$                      ;IF NOT, UNRECOGNIZED RESPONSE
    004707  050000  004721              ^050000,6$
 41 004711                              CALL    STOSTA                  ;CHECK STATUS
    004711  020000  005437              ^020000,STOSTA
 42 004713                              ERRHRD  MS2026                  ;
    004713  020000  006261              ^020000,RERROR
    004715  103752                                                              .WORD ERHARD+ERRN
    004716  001712                                                              .WORD <PRMS*10000>+MS2026
 43 004717                              BR      4$
    004717  000000  004742              ^00,4$
 44 004721                      6$:     ERRHRD  MS2027,#RDM.EN,R0
    004721  100467                                                              MOV R0,-(SP)
    004722  104010  001624                                                      MOV R1,SAVREG
    004724  104201  000162                                                      MOV #RDM.EN,R1
    004726  100461                                                              MOV R1,-(SP)
    004727  104301  001624                                                      MOV SAVREG,R1
    004731  020000  006261              ^020000,RERROR
    004733  103753                                                              .WORD ERHARD+ERRN
    004734  021737                                                              .WORD <PRMS*10000>+MS2027
 45 004735                              BR      4$                      ;ERROR EXIT
    004735  000000  004742              ^00,4$
 46 004737  114002                 3$:  CLR     R2                      ;ALL OK, EXIT
 47 004740                              BR      5$                      ;
    004740  000000  004744              ^00,5$
 48 004742  104202  000001          4$: MOV     #1,R2                   ;ERROR OCCURED, SET R2
 49 004744                          5$: POP     <R3,R1,R0>
    004744  104263                                                              MOV (SP)+,R3
    004745  104261                                                              MOV (SP)+,R1
    004746  104267                                                              MOV (SP)+,R0
 50 004747                              RETURN
    004747  000000  000000              ^00,0
```

```
                                        .SBTTL  WRITE MEMORY SUBROUTINE
                                ;:+
                                ;       WRTMEM
                                ;
                                ;       DESCRIPTION:
                                ;       THIS ROUTINE DOES THE XFC WRITE MEMORY TO A SPECIFIED
                                ;       REGION ID AND OFFSET WITH A BYTE COUNT AND DATA.
                                ;       THIS ROUTINE CALLS THE SEND AND RCV (TALKER) ROUTINE
                                ;       AND JUDGES IF THE SDI COMMAND WAS PROPERLY SENT.
                                ;
                                ;       INPUT:  ALL PARAMETERS FOR THE WRITE MEMORY COMMAND MUST BE SET.
                                ;               REGION ID, OFFSET, BYTE COUNT AND THE DATA MUST BE STORED
                                ;               IN THE PACKET.
                                ;
                                ;       OUTPUT: R2 = 0 MEANS WRITE MEMORY COMMAND WAS PROCESSED SUCCESSFULLY
                                ;               R2 NOT = 0, WRITE FAILED (EITHER SEND OR RECEIVE OF THE COMMAND)
                                ;:-
004751  104203  001657        WRTMEM: MOV     #CR.WRM,R3              ;R3->MEM WRITE PACKET
004753  104302  001412                MOV     SDI,R2                  ;R2 = PORT
004755                               CALL    TALKER                  ;SEND COMMAND AND RECEIVE RESPONSE FROM DRIVE
004755  020000  006131                ^O020000,TALKER
004757  115003                       TST     R3                      ;SEE IF ERROR OCCURRED
004760                               BEQ     2$                      ;IF NO ERRORS, BRANCH
004760  010000  005005                ^O010000,2$
004762  110203                       ROL     R3                      ;SHIFT HIGH BIT INTO CARRY BIT
004763                               BCC     1$                      ;IF CARRY CLEAR (RECEIVE ERROR) BRANCH
004763  040000  004773                ^O040000,1$
                                ; *** REPORT TRANSMISSION ERROR
004765                               ERRHRD  MS2028
004765  020000  006261                ^O020000,RERROR
004767  103754                                                       .WORD ERHARD+ERRN
004770  002027                                                       .WORD <PRMS*10000>+MS2028
004771                               BR      4$
004771  000000  005046                ^O00,4$
                                ; *** REPORT RECEPTION ERROR
004773                          1$:  CALL    TYPERR                  ;FIND OUT WHAT TYPE OF ERROR OCCURED
004773  020000  005674                ^O020000,TYPERR
004775                               ERRHRD  MS2029,R0,R3
004775  100463                                                       MOV R3,-(SP)
004776  100467                                                       MOV R0,-(SP)
004777  020000  006261                ^O020000,RERROR
005001  103755                                                       .WORD ERHARD+ERRN
005002  022062                                                       .WORD <PRMS*10000>+MS2029
005003                               BR      4$
005003  000000  005046                ^O00,4$
                                ; *** SET UP RESPONSE PACKET
005005  106207  000176          2$:  CMP     #COMPLT,R0              ;DID IT COMPLETE
005007                               BEQ     3$                      ;IF SO CONTINUE
005007  010000  005043                ^O010000,3$
005011  106207  000175                CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
005013                               BNE     6$                      ;IF NOT, UNRECOGNIZED RESPONSE
005013  050000  005025                ^O050000,6$
005015                               CALL    STOSTA                  ;CHECK STATUS
005015  020000  005437                ^O020000,STOSTA
005017                               ERRHRD  MS2030                  ;
005017  020000  006261                ^O020000,RERROR
005021  103756                                                       .WORD ERHARD+ERRN
```

```
      005022  002121                                                        .WORD <PRMS*10000>+MS2030
   39 005023                          BR      4$
      005023  000000  005046          ^O0,4$
   40 005025                  6$:     ERRHRD  MS2031,#COMPLT,R0                MOV R0,-(SP)
      005025  100467                                                        MOV R1,SAVREG
      005026  104010  001624                                                MOV #COMPLT,R1
      005030  104201  000176                                                MOV R1,-(SP)
      005032  100461                                                        MOV SAVREG,R1
      005033  104301  001624
      005035  020000  006261          ^O020000,RERROR                       .WORD ERHARD+ERRN
      005037  103757                                                        .WORD <PRMS*10000>+MS2031
      005040  022147
   41 005041                          BR      4$                  ;ERROR EXIT
      005041  000000  005046          ^O0,4$
   42 005043  114002                  3$:     CLR     R2                  ;ALL OK, EXIT
   43 005044                          BR      5$                  ;
      005044  000000  005050          ^O0,5$
   44 005046  104202  000001          4$:     MOV     #1,R2               ;ERROR OCCURED, SET R2
   45 005050                          5$:     RETURN
      005050  000000  000000          ^O0,0
```

RUN (OR SPIN UP) COMMAND

```
                                        .SBTTL  RUN (OR SPIN UP) COMMAND
                                ;ISSUE RUN COMMAND AND CHECK THAT IT PERFORMS PROPERLY

005052  104203  002136        RUNDR:  MOV     #CR.RUN,R3              ;R3 -> COMMAND
005054                                  CALL    TALKER                  ;SEND TO DRIVE
005054  020000  006131                  ^O020000,TALKER
005056  115003                          TST     R3                      ;ALL OK?
005057                                  BEQ     60$                     ;IF SO, CHECK RESPONCE CODE
005057  010000  005103                  ^O010000,60$
005061                                  BPL     50$                     ;ELSE, IS MSB SET?/IF NOT, RECEIVE ERROR
005061  030000  005071                  ^O030000,50$
005063                                  ERRHRD  MS2032                  ;REPORT SEND ERROR
005063  020000  006261                  ^O020000,RERROR
005065  103760                                                                  .WORD ERHARD+ERRN
005066  002237                                                                  .WORD <PRMS*10000>+MS2032
005067                                  BR      70$                     ;
005067  000000  005137                  ^O00,70$
005071                          50$:    CALL    TYPERR
005071  020000  005674                  ^O020000,TYPERR
005073                                  ERRHRD  MS2033,R0,R3            ;REPORT RECEIVE ERROR
005073  100463                                                                  MOV R3,-(SP)
005074  100467                                                                  MOV R0,-(SP)
005075  020000  006261                  ^O020000,RERROR
005077  103761                                                                  .WORD ERHARD+ERRN
005100  022265                                                                  .WORD <PRMS*10000>+MS2033
005101                                  BR      70$                     ;
005101  000000  005137                  ^O00,70$
005103  106207  000176          60$:    CMP     #COMPLT,R0              ;CHECK RESPONSE CODE
005105                                  BEQ     75$                     ;IF OK, CONTINUE
005105  010000  005143                  ^O010000,75$
005107  106207  000175                  CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
005111                                  BNE     65$                     ;IF NOT, UNRECOGNIZED RESPONSE
005111  050000  005123                  ^O050000,65$
005113                                  CALL    STOSTA                  ;CHECK STATUS
005113  020000  005437                  ^O020000,STOSTA
005115                                  ERRHRD  MS2034                  ;
005115  020000  006261                  ^O020000,RERROR
005117  103762                                                                  .WORD ERHARD+ERRN
005120  002317                                                                  .WORD <PRMS*10000>+MS2034
005121                                  BR      70$
005121  000000  005137                  ^O00,70$
005123                          65$:    ERRHRD  MS2035,#COMPLT,R0       ;IF NOT, REPORT ERROR
005123  100467                                                                  MOV R0,-(SP)
005124  104010  001624                                                          MOV R1,SAVREG
005126  104201  000176                                                          MOV #COMPLT,R1
005130  100461                                                                  MOV R1,-(SP)
005131  104301  001624                                                          MOV SAVREG,R1
005133  020000  006261                  ^O020000,RERROR
005135  103763                                                                  .WORD ERHARD+ERRN
005136  022340                                                                  .WORD <PRMS*10000>+MS2035
005137  104202  000001          70$:    MOV     #1,R2                   ;R2 IS NOT ZERO MEANING ERROR
005141                                  BR      80$
005141  000000  005144                  ^O00,80$
005143  114002                  75$:    CLR     R2                      ;R2 IS ZERO, NOT ERROR
005144                          80$:    RETURN
005144  000000  000000                  ^O00,0
```

```
                                        .SBTTL  RECALIBRATE COMMAND
                                ;ISSUE RECALIBRATE COMMAND AND CHECK THAT IT PERFORMS PROPERLY

005146  104203  002144        RECAL:  MOV     #CR.INR,R3              ;R3 -> COMMAND
005150                                CALL    TALKER                  ;SEND TO DRIVE
005150  020000  006131                ^O020000,TALKER
005152  115003                        TST     R3                      ;ALL OK?
005153                                BEQ     90$                     ;IF SO, CHECK RESPONCE CODE
005153  010000  005177                ^O010000,90$
005155                                BPL     80$                     ;ELSE, IS MSB SET?/IF NOT, RECEIVE ERROR
005155  030000  005165                ^O030000,80$
005157                                ERRHRD  MS2036                  ;REPORT SEND ERROR
005157  020000  006261                ^O020000,RERROR
005161  103764                                                                .WORD ERHARD+ERRN
005162  002424                                                                .WORD <PRMS*10000>+MS2036
005163                                BR      100$                    ;
005163  000000  005233                ^O00,100$
005165                          80$:  CALL    TYPERR
005165  020000  005674                ^O020000,TYPERR
005167                                ERRHRD  MS2037,R0,R3            ;REPORT RECEIVE ERROR
005167  100463                                                                MOV R3,-(SP)
005170  100467                                                                MOV R0,-(SP)
005171  020000  006261                ^O020000,RERROR
005173  103765                                                                .WORD ERHARD+ERRN
005174  022456                                                                .WORD <PRMS*10000>+MS2037
005175                                BR      100$                    ;
005175  000000  005233                ^O00,100$
005177  106207  000176          90$:  CMP     #COMPLT,R0              ;CHECK RESPONSE CODE
005201                                BEQ     105$                    ;IF OK, CONTINUE
005201  010000  005237                ^O010000,105$
005203  1062^^  000175                CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
005205                                BNE     95$                     ;IF NOT, UNRECOGNIZED RESPONSE
005205  050000  005217                ^O050000,95$
005207                                CALL    STOSTA                  ;CHECK STATUS
005207  020000  005437                ^O020000,STOSTA
005211                                ERRHRD  MS2038                  ;
005211  020000  006261                ^O020000,RERROR
005213  103766                                                                .WORD ERHARD+ERRN
005214  002514                                                                .WORD <PRMS*10000>+MS2038
005215                                BR      100$
005215  000000  005233                ^O00,100$
005217                          95$:  ERRHRD  MS2039,#COMPLT,R0       ;IF NOT, REPORT ERROR
005217  100467                                                                MOV R0,-(SP)
005220  104010  001624                                                        MOV R1,SAVREG
005222  104201  000176                                                        MOV #COMPLT,R1
005224  100461                                                                MOV R1,-(SP)
005225  104301  001624                                                        MOV SAVREG,R1
005227  020000  006261                ^O020000,RERROR
005231  103767                                                                .WORD ERHARD+ERRN
005232  022541                                                                .WORD <PRMS*10000>+MS2039
005233  104202  000001          100$:  MOV     #1,R2
005235                                BR      110$
005235  000000  005240                ^O00,110$
005237  114002                  105$:  CLR     R2
005240                          110$:  RETURN
005240  000000  000000                ^O00,0
```

```
                                        .SBTTL  GET STATUS SUBROUTINE
                                ;
                                ;       GET STATUS
                                ;
                                ;       OUTPUT  R3 = 0 IF OK
                                ;               R3 = 1 IF ERROR
                                ;               R4 USED AS A DECREMENT COUNTER
                                ;
 005242  114004                 GTSTAT: CLR     R4
 005243  104203  001642         3$:     MOV     #CR.GST,R3              ;POINT TO GET STATUS COMMAND
 005245                                 CALL    TALKER                  ; SEND AND RECEIVE COMMAND
 005245  020000  006131                 ^O020000,TALKER
 005247  115003                         TST     R3                      ; SEE IF ERROR OCCURED
 005250                                 BEQ     2$                      ; IF NOT, BRANCH
 005250  010000  005300                 ^O010000,2$
 005252  110203                         ROL     R3                      ; SEE IF RECEIVE ERROR
 005253                                 BCC     1$                      ; IF SO, BRANCH
 005253  040000  005266                 ^O040000,1$
 005255  117404                         DEC     R4                      ; IF SEND ERROR, COUNTER DONE?
 005256                                 BNE     3$                      ; IF NOT, TRY AGAIN
 005256  050000  005243                 ^O050000,3$
 005260                                 ERRHRD  MS2040
 005260  020000  006261                 ^O020000,RERROR
 005262  103770                                                                 .WORD ERHARD+ERRN
 005263  002631                                                                 .WORD <PRMS*10000>+MS2040
 005264                                 BR      ER.END
 005264  000000  005334                 ^O00,ER.END
 005266                         1$:     CALL    TYPERR
 005266  020000  005674                 ^O020000,TYPERR
 005270                                 ERRHRD  MS2041,R0,R3
 005270  100463                                                                 MOV R3,-(SP)
 005271  100467                                                                 MOV R0,-(SP)
 005272  020000  006261                 ^O020000,RERROR
 005274  103771                                                                 .WORD ERHARD+ERRN
 005275  022663                                                                 .WORD <PRMS*10000>+MS2041
 005276                                 BR      ER.END
 005276  000000  005334                 ^O00,ER.END
                                ; *** LOOK AT RESPONSE OP-CODE FROM DRIVE -- SHOULD BE DATA RESPONSE
 005300  106207  000366         2$:     CMP     #STSRES,R0              ;CHECK OP-CODE
 005302                                 BEQ     OK.END                  ;BRANCH IF A MATCH
 005302  010000  005340                 ^O010000,OK.END
 005304  106207  000175                 CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
 005306                                 BNE     4$                      ;IF NOT, UNRECOGNIZED RESPONSE
 005306  050000  005320                 ^O050000,4$
 005310                                 CALL    STOSTA
 005310  020000  005437                 ^O020000,STOSTA
 005312                                 ERRHRD  MS2042                  ;
 005312  020000  006261                 ^O020000,RERROR
 005314  103772                                                                 .WORD ERHARD+ERRN
 005315  002721                                                                 .WORD <PRMS*10000>+MS2042
 005316                                 BR      ER.END
 005316  000000  005334                 ^O00,ER.END
 005320                         4$:     ERRHRD  MS2043,#STSRES,R0       ;WRONG RESPONSE FROM DRIVE
 005320  000467                                                                 MOV R0,-(SP)
 005321  004010  001624                                                         MOV R1,SAVREG
 005323  104201  000366                                                         MOV #STSRES,R1
 005325  100461                                                                 MOV R1,-(SP)
```

```
        005326  104301  001624                                                  MOV SAVREG,R1
        005330  020000  006261                ^O20000,RERROR
        005332  103773                                                          .WORD ERHARD+ERRN
        005333  022746                                                          .WORD <PRMS*10000>+MS2043
     32 005334  104203  000001        ER.END: MOV     #1,R3           ;R3 = NONE ZERO VALUE WHEN DONE(ERROR)
     33 005336                              BR      EX.END          ;EXIT
        005336  000000  005341              ^O0,EX.END
     34 005340  114003                OK.END: CLR     R3              ;R3 = 0 WHEN DONE (NO ERROR)
     35 005341                        EX.END: RETURN
        005341  000000  000000              ^O0,0
```

```
                                        .SBTTL  CLEAR DRIVE SUBROUTINE
                                ;
                                ;       DRIVE CLEAR
                                ;
                                ;       OUTPUT  R3 = 0 IF OK
                                ;               R3 = 1 IF ERROR
                                ;               R4 USED AS A DECREMENT COUNTER
                                ;
005343  114004          CLRDRV: CLR     R4                      ; CLEAR DECEMENT COUNTER
005344  104203  001650  3$:     MOV     #CR.DRC,R3              ; POINT TO DRIVE CLEAR COMMAND
005346                          CALL    TALKER                  ; SEND AND RECEIVE COMMAND
005346  020000  006131          ^O020000,TALKER
005350  115003                  TST     R3                      ; SEE IF ERROR OCCURED
005351                          BEQ     2$                      ; IF NOT, BRANCH
005351  010000  005401          ^O010000,2$
005353  110203                  ROL     R3                      ; SEE IF RECEIVE ERROR
005354                          BCC     1$                      ; IF SO, BRANCH
005354  040000  005367          ^O040000,1$
005356  117404                  DEC     R4                      ; ON SEND ERPOR, IS COUNTER DONE?
005357                          BNE     3$                      ; IF NOT, TRY AGAIN
005357  050000  005344          ^O050000,3$
005361                          ERRHRD  MS2044
005361  020000  006261          ^O020000,RERROR
005363  103774                                                  .WORD ERHARD+ERRN
005364  003035                                                  .WORD <PRMS*10000>+MS2044
005365                          BR      ER.END
005365  000000  005334          ^O0,ER.END
005367                  1$:     CALL    TYPERR
005367  020000  005674          ^O020000,TYPERR
005371                          ERRHRD  MS2045,R0,R3
005371  100463                                                  MOV R3,-(SP)
005372  100467                                                  MOV R0,-(SP)
005373  020000  006261          ^O020000,RERROR
005375  103775                                                  .WORD ERHARD+ERRN
005376  023067                                                  .WORD <PRMS*10000>+MS2045
005377                          BR      ER.END
005377  000000  005334          ^O0,ER.END
                        ; *** CHECK RESPONSE CODE
005401  106207  000176  2$:     CMP     #COMPLT,R0              ; SEE IF CORRECT RESPONSE CODE RECEIVED
005403                          BEQ     OK.END                  ; IF SUCCESSFUL, BRANCH
005403  010000  005340          ^O010000,OK.END
005405  106207  000175          CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
005407                          BNE     4$                      ;IF NOT, UNRECOGNIZED RESPONSE
005407  050000  005421          ^O050000,4$
005411                          CALL    STOSTA                  ;CHECK STATUS
005411  020000  005437          ^O020000,STOSTA
005413                          ERRHRD  MS2046                  ;
005413  020000  006261          ^O020000,RERROR
005415  103776                                                  .WORD ERHARD+ERRN
005416  003125                                                  .WORD <PRMS*10000>+MS2046
005417                          BR      ER.END
005417  000000  005334          ^O0,ER.END
005421                  4$:     ERRHRD  MS2047,#COMPLT,R0       ;CLEAR ERROR FAILED
005421  100467                                                  MOV R0,-(SP)
005422  104010  001624                                          MOV R1,SAVREG
005424  104201  000176                                          MOV #COMPLT,R1
005426  100461                                                  MOV R1,-(SP)
```

```
        005427  104301  001624                                  MOV SAVREG,R1
        005431  020000  006261          ^O20000,RERROR
        005433  103777                                          .WORD ERHARD+ERRN
        005434  023152                                          .WORD <PRMS*10000>+MS2047
     32 005435                          BR      ER.END
        005435  000000  005334          ^O0,ER.END
     33
```

```
                                        .SBTTL  STORE STATUS ROUTINE
                                ;       STORE STATUS
                                ;
                                ;               MOVE WHAT DATA IS IN ST AND PUT IT IN OUTPUT BUFFER
                                STOSTA:
                                        PUSH    <R0,R1,R2>
                                                                                        MOV R0,-(SP)
                                                                                        MOV R1,-(SP)
                                                                                        MOV R2,-(SP)
                                        MOV     #OUT.05,R0              ;R0 -> OUTPUT BUFFER
                                        MOV     #STAT0,R1               ;MOVE FORMAT ADDRESS IN OUTPUT BUFFER
                                        MOV     R1,(R0)+
                                        MOV     SDI,R2                  ; R2 PORT INDICATOR
                                        XFC     STATUS                  ; GET STATUS
                                        BIC     #^C100507,R1            ; CLEAR UNUSED BITS
                                        MOV     R1,(R0)+                ; STORE
                                        MOV     #ST+7,R1                ;R1 -> INPUT BUFFER
                                1$:     MOV     -(R1),R2                ;SAVE
                                        MOV     R2,(R0)+
                                        CMP     #ST,R1                  ;DONE?
                                        BNE     1$                      ;IF NOT, CONTINUE
                                        ^O50000,1$
                                        POP     <R2,R1,R0>
                                                                                        MOV (SP)+,R2
                                                                                        MOV (SP)+,R1
                                                                                        MOV (SP)+,R0
                                        RETURN
                                        ^O0,0
```

```
                                        .SBTTL  CONVERT MEMORY -- SKEWED BY BYTE
                                ;+
                                ;       ROUTINE NAME:   CONMEM
                                ;
                                ;       DESCRIPTION:
                                ;       MEMORY IS SENT TO THE DM ROUTINE STARTING ON THE ODD BYTE BOUNDARY.
                                ;       THIS ROUTINE SHIFTS THE DATA TO START ON THE EVEN BYTE BOUNDARY AND
                                ;       STORES EACH WORD INTO A SPECIFIED BUFFER AREA.  THE DATA IS STORED AS
                                ;       FOLLOWS:
                                ;               ST:     DATA BYTE 0     BYTE COUNT
                                ;                       DATA BYTE 2     DATA BYTE 1
                                ;                       DATA BYTE 4     DATA BYTE 3
                                ;                       .....           DATA BYTE 5
                                ;
                                ;       AFTER EXECUTION, THD DATA IS STORED LIKE THIS:
                                ;               OUT.XX: DATA BYTE 1     DATA BYTE 0
                                ;                       DATA BYTE 3     DATA BYTE 2
                                ;                       DATA BYTE 5     DATA BYTE 4
                                ;                       .....           DATA BYTE 6
                                ;
                                ;       INPUT:  R4 -> BUFFER AREA'S FIRST LOCATION
                                ;               ST HAS INPUT DATA
                                ;
                                ;       OUTPUT: OUTPUT BUFFER HAS SHIFTED DATA
                                ;
                                ;-
005472                          CONMEM: PUSH    <R0,R1,R2,R3,R4>
005472  100467                                                          MOV R0,-(SP)
005473  100461                                                          MOV R1,-(SP)
005474  100462                                                          MOV R2,-(SP)
005475  100463                                                          MOV R3,-(SP)
005476  100464                                                          MOV R4,-(SP)
005477  104203  001415                  MOV     #ST,R3                 ;R3 -> BYTE COUNT
005501  104137                          MOV     (R3),R0                ;R0 = BYTE COUNT
005502  103207  177400                  BIC     #HIBYTE,R0             ;STRIP OFF EXTRANEOUS
005504  104231                  1$:     MOV     (R3)+,R1               ;R1 = EVEN BYTE OF DATA
005505  103201  000377                  BIC     #LOBYTE,R1             ;STRIP OFF EXTRANEOUS
005507  117407                          DEC     R0                     ;DONE?
005510                                  BEQ     2$                     ;IF YES, GO STORE LAST BYTE
005510  010000  005525                  ^O10000,2$
005512  104132                          MOV     (R3),R2                ;R2 = ODD BYTE
005513  103202  177400                  BIC     #HIBYTE,R2             ;STRIP OFF EXTRANEOUS
005515  101021                          BIS     R2,R1                  ;R1 HAS WHOLE WORD(BYTES REVERSED)
005516  110701                          SWAB    R1                     ;SWITCH BYTES
005517  100241                          MOV     R1,(R4)+               ;STORE DATA
005520  117407                          DEC     R0                     ;DONE?
005521                                  BEQ     3$                     ;IF SO, EXIT(EVEN # OF BYTES)
005521  010000  005527                  ^O10000,3$
005523                                  BR      1$                     ;ELSE, CONTINUE
005523  000000  005504                  ^O0,1$
005525  110701                  2$:     SWAB    R1                     ;ODD # BYTES TO GET HERE
005526  100141                          MOV     R1,(R4)                ;STORE LAST BYTE
005527                          3$:     POP     <R4,R3,R2,R1,R0>
005527  104264                                                          MOV (SP)+,R4
005530  104263                                                          MOV (SP)+,R3
005531  104262                                                          MOV (SP)+,R2
005532  104261                                                          MOV (SP)+,R1
```

```
        005533  104267                                          MOV (SP)+,R0
     46 005534                          RETURN
        005534  000000  000000          ^O0,0
```

```
                                .SBTTL  GET BYTE COUNT
                        ;+
                        ;       ROUTINE NAME:   GETBCN
                        ;
                        ;       DESCRIPTION:
                        ;       DATA IS READ BY THE DM PROGRAM AFTER THE DIAGNOSE COMMAND.
                        ;       WITHIN THIS DATA, THE DRIVE SPECIFIES IF MORE DATA NEEDS TO
                        ;       BE READ.  IF IT IS, THE BYTE COUNT MUST BE CONVERTED FROM
                        ;       ITS PRESENT FORM WHICH FOLLOWS THIS FORMAT:
                        ;
                        ;       BIT 15                                          BIT 0
                        ;       +-----------------------+-----------------------+
                        ;OUT.08:!       ASCII COUNT     !     BINARY COUNT      !
                        ;       +-----------------------+-----------------------+
                        ;
                        ;       THE TOTAL BYTE COUNT IS FOUND BY THIS FORMULA:
                        ;                       TBC = AC + (4*BC)
                        ;       OR THE TOTAL BYTE COUNT EQUALS THE ASCII COUNT PLUS FOUR
                        ;       TIMES THE BINARY COUNT.  THE BINARY COUNT IS THE NUMBER OF
                        ;       32 BIT BINARY VALUES INCLUDED IN THE REPORT.  THE ASCII
                        ;       COUNT IS THE NUMBER OF ASCII CHARACTERS INCLUDED IN THE
                        ;       REPORT.
                        ;
                        ;       INPUT:  OUT.08 = ASCII COUNT AND BINARY COUNT IN THE FORM STATED ABOVE
                        ;
                        ;       OUTPUT: RDM+3 HAS UPDATED BYTE COUNT.
                        ;
                        ;-
005536                  GETBCN: PUSH    <R0,R1,R2,R3,R4,R5>
005536  100467                                                          MOV R0,-(SP)
005537  100461                                                          MOV R1,-(SP)
005540  100462                                                          MOV R2,-(SP)
005541  100463                                                          MOV R3,-(SP)
005542  100464                                                          MOV R4,-(SP)
005543  100465                                                          MOV R5,-(SP)
005544  114007                          CLR     R0
005545  110207                          ROL     R0                      ;CLEAR CARRY
005546  104307  002162                  MOV     OUT.08,R0               ;GET BYTE COUNT
005550  103207  177400                  BIC     #HIBYTE,R0              ;R0 = BINARY COUNT= # 32 BIT WORDS
005552  110207                          ROL     R0                      ;
005553  110207                          ROL     R0                      ;R0 = BINARY COUNT IN BYTES
005554  104301  002162                  MOV     OUT.08,R1               ;R1 = BYTE COUNT
005556  110701                          SWAB    R1                      ;ASCII COUNT IN LO BYTE
005557  103201  177400                  BIC     #HIBYTE,R1              ;R1 = ASCII COUNT
005561  104012                          MOV     R1,R2                   ;SAVE IN R2 = ASCII COUNT
005562  105017                          ADD     R1,R0                   ;R0 = TOTAL BYTE COUNT
                                ; *** IS SIZE OF THE PACKET TOO BIG FOR BUFFER AVAILABLE?
005563  104201  000052                  MOV     #42.,R1                 ;42. IS THE MAXIMUM # OF BYTES LEFT
005565  106017                          CMP     R1,R0                   ;IS TOTAL BYTE COUNT > 42.?
005566                                  BPL     5$                      ;IF NOT, GO STORE TOTAL BYTE COUNT
005566  030000  005662                  ^O030000,5$
                                ; *** SAVE OUT.RQ DATA
005570  104204  002153                  MOV     #OUT.01,R4              ;R4-> OUT.RQ DATA
005572  104205  001555                  MOV     #ST+140,R5              ;R5->SAVE AREA
005574  104241                  1$:     MOV     (R4)+,R1                ;SAVE
005575  100251                          MOV     R1,(R5)+                ;
005576  106204  002214                  CMP     #OUT.34,R4              ;DONE?
```

GET BYTE COUNT

```
 51 005600                              BNE     1$                      ;IF NOT, CONTINUE
    005600  050000  005574              ^050000,1$
 52                             ;
 53 005602                              MSSGE   MSG2
    005602  104300  001623  002153                      MOV     LUNIT,OUT.01
    005605  104200  003342  002154                      MOV     #MSG2,OUT.02
    005610  100467                                              MOV R0,-(SP)
    005611  100461                                              MOV R1,-(SP)
    005612  104207  060015                              MOV     #MESSAG,R0
    005614  020000  006066              ^020000,HOSTRQ
    005616  104261                                              MOV (SP)+,R1
    005617  104267                                              MOV (SP)+,R0
 54                             ; *** RESTORE OUT.RQ DATA
 55 005620  104204  002153              MOV     #OUT.01,R4      ;R4-> OUT.RQ DATA
 56 005622  104205  001555              MOV     #ST+140,R5      ;R5->SAVE AREA
 57 005624  104251              2$:     MOV     (R5)+,R1        ;RESTORE
 58 005625  100241                      MOV     R1,(R4)+        ;
 59 005626  106204  002214              CMP     #OUT.34,R4      ;DONE?
 60 005630                              BNE     2$              ;IF NOT, CONTINUE
    005630  050000  005624              ^050000,2$
 61                             ;
 62 005632  104073                      MOV     R0,R3           ;SAVE ASCII COUNT + 4*BINARY COUNT IN R3
 63 005633  104207  000052              MOV     #42.,R0         ;ELSE, MAXIMUM BYTE COUNT IS STORED.
 64                             ; *** ADJUST ASCII COUNT IF TOO LARGE
 65 005635  107023                      SUB     R2,R3           ;R3 = BINARY COUNT
 66 005636  106073                      CMP     R0,R3           ;IF BINARY COUNT ALONE > 42.?
 67 005637                              BMI     3$              ;IF SO, TO CLEAR ASCII COUNT AND ADJUST BINARY COUNT
    005637  070000  005650              ^070000,3$
 68                             ; *** ASCII COUNT IS > 42. TO GET HERE
 69 005641  105032                      ADD     R3,R2           ;R2 = TOTAL BYTE COUNT
 70 005642  107072                      SUB     R0,R2           ;R2 = ASCII COUNT ADJUSTMENT
 71 005643  110702                      SWAB    R2              ;PUT ASCII COUNT ADJUSTMENT IN UPPER BYTE
 72 005644  107020  002162              SUB     R2,OUT.08       ;SUBTRACT FROM OUT.08 FOR HOST
 73 005646                              BR      5$              ;EXIT
    005646  000000  005662              ^00,5$
 74                             ; *** ADJUST BINARY COUNT IF TOO LARGE/CLEAR OUT ASCII COUNT
 75 005650  103200  177400  002162 3$:     BIC     #HIBYTE,OUT.08  ;CLEAR HI BYTE OF OUT.08
 76 005653  107073                      SUB     R0,R3           ;R3 = BINARY COUNT ALONE
 77 005654  117400  002162      4$:     DEC     OUT.08          ;ADJUST BINARY COUNT
 78 005656  107203  000004              SUB     #4,R3
 79 005660                              BPL     4$
    005660  030000  005654              ^030000,4$
 80                             ; *** STORE IN READ MEMORY PACKET AND EXIT
 81 005662                      5$:
 82 005662  104070  002103              MOV     R0,RDM+3        ;STORE IN READ MEMORY PACKET
 83 005664                              POP     <R5,R4,R3,R2,R1,R0>
    005664  104265                                                      MOV (SP)+,R5
    005665  104264                                                      MOV (SP)+,R4
    005666  104263                                                      MOV (SP)+,R3
    005667  104262                                                      MOV (SP)+,R2
    005670  104261                                                      MOV (SP)+,R1
    005671  104267                                                      MOV (SP)+,R0
 84 005672                              RETURN
    005672  000000  000000              ^00,0
```

```
                                                .SBTTL  TYPE WHAT KIND OF RECEIVE ERROR
                                        ;;+
                                        ;       TYPE ERROR
                                        ;
                                        ;       DESCRIPTION:
                                        ;       THIS ROUTINE PRINTS A REPORT TO TELL WHAT KIND OF ERROR OCCURED IF A
                                        ;       RECEIVE XFC DID NOT SUCCESSFULLY COMPLETE.
                                        ;
                                        ;       INPUT:  R3 = ERROR VALUE FROM THE XFC SAVED IN TALKER ROUTINE SHIFTED LEFT ONCE
                                        ;                       IF = 1 TIMEOUT
                                        ;                       IF = 2 1ST WORD WAS NOT A START FRAME
                                        ;                       IF = 4 FRAMING ERROR ON SDI LEVEL 0 READ
                                        ;                       IF = 10 CHECKSUM ERROR ON SDI LEVEL 0 READ
                                        ;                       IF = 20 BUFFER SIZE WAS SMALLER THAN RESPONSE
                                        ;;-
005674  110603                 TYPERR: ROR     R3                      ;READJUST R3
005675  114001                         CLR     R1
005676  110201                         ROL     R1                      ;CLEAR CARRY
005677  104031                         MOV     R3,R1                   ;STORE IN R1
005700  110601                         ROR     R1                      ;ERROR?
005701                                 BCC     1$                      ;IF NOT, CONTINUE
005701  040000  005707                 ^040000,1$
005703  104207  003374                 MOV     #MSR0,R0                ;TIMEOUT
005705                                 BR      6$                      ;EXIT
005705  000000  005745                 ^00,6$
005707  110601                 1$:     ROR     R1                      ;ERROR?
005710                                 BCC     2$                      ;IF NOT, CONTINUE
005710  040000  005716                 ^040000,2$
005712  104207  003423                 MOV     #MSR1,R0                ;1ST WORD WAS NOT A START FRAME
005714                                 BR      6$                      ;EXIT
005714  000000  005745                 ^00,6$
005716  110601                 2$:     ROR     R1                      ;ERROR?
005717                                 BCC     3$                      ;IF NOT, CONTINUE
005717  040000  005725                 ^040000,3$
005721  104207  003453                 MOV     #MSR2,R0                ;FRAMING ERROR ON SDI LEVEL 0 READ
005723                                 BR      6$                      ;EXIT
005723  000000  005745                 ^00,6$
005725  110601                 3$:     ROR     R1                      ;ERROR?
005726                                 BCC     4$                      ;IF NOT, CONTINUE
005726  040000  005734                 ^040000,4$
005730  104207  003514                 MOV     #MSR3,R0                ;CHECKSUM ERROR ON SDI LEVEL 0 READ
005732                                 BR      6$                      ;EXIT
005732  000000  005745                 ^00,6$
005734  110601                 4$:     ROR     R1                      ;ERROR?
005735                                 BCC     5$                      ;IF NOT, CONTINUE
005735  040000  005743                 ^040000,5$
005737  104207  003555                 MOV     #MSR4,R0                ;BUFFER SIZE SMALLER THAN RESPONSE
005741                                 BR      6$                      ;EXIT
005741  000000  005745                 ^00,6$
005743  104207  003612         5$:     MOV     #MSR5,R0                ;ERROR WASN'T PROPERLY SPECIFIED
005745                         6$:     RETURN
005745  000000  000000                 ^00,0
```

```
                                        .SBTTL  DIVIDE BY OCTAL 50 AND FIND ASCII EQUIVALENT
                                ;;+
                                ;       DIVIDE BY 50
                                ;
                                ;       DESCRIPTION:    RAD 50 VALUE IN R4 IS DIVIDED BY 50(OCTAL)
                                ;                       TO STRIP OFF A SINGLE CHARACTER INTO R0.
                                ;                       JUMP TO FNDASC TO FIND THE ASCII EQUIVALENT
                                ;
                                ;       INPUT:  R4 HAS RAD50 VALUE
                                ;
                                ;       OUTPUT: R0 HAS ASCII CHARACTER
                                ;;-
005747                          DIV50:  PUSH    <R1>
005747  100461                                                                  MOV R1,-(SP)
005750  114001                          CLR     R1                      ;R1 = QUO
005751  115401                  1$:     INC     R1                      ;KEEP TRACK OF # OF LOOPS
005752  107204  000050                  SUB     #50,R4                  ;DONE?
005754                                  BPL     1$                      ;IF NOT, LOOP
005754  030000  005751                  ^O030000,1$
005756  105204  000050                  ADD     #50,R4                  ;R4 = REMAINDER
005760  104047                          MOV     R4,R0                   ;STORE REMAINDER IN R0
005761  117401                          DEC     R1                      ;R1 HAS 1 MORE THAN IT SHOULD
005762  104014                          MOV     R1,R4                   ;QUO IN R4
005763                                  POP     <R1>
005763  104261                                                                  MOV (SP)+,R1

                                ;;+
                                ;       FIND ASCII
                                ;
                                ;       INPUT:  R0 HAS RAD 50 CHARACTER
                                ;;-
005764  115007                  FNDASC: TST     R0                      ;IS = 0?
005765                                  BNE     2$                      ;IF NOT, CONTINUE
005765  050000  005773                  ^O050000,2$
005767  104207  000040                  MOV     #40,R0                  ;IF SO, SPACE SET
005771                                  RETURN
005771  000000  000000                  ^O0,0
005773  106207  000032          2$:     CMP     #32,R0                  ; > LARGEST LETTER
005775                                  BMI     3$                      ;IF SO, CONTINUE
005775  070000  006003                  ^O070000,3$
005777  105207  000100                  ADD     #100,R0                 ;SET ASCII LETTER
006001                                  RETURN
006001  000000  000000                  ^O0,0
006003  106207  000033          3$:     CMP     #33,R0                  ; = 33?
006005                                  BNE     4$                      ;IF NOT, CONTINUE
006005  050000  006013                  ^O050000,4$
006007  104207  000044                  MOV     #44,R0                  ;SET '$'
006011                                  RETURN
006011  000000  000000                  ^O0,0
006013  106207  000034          4$:     CMP     #34,R0                  ; = 34?
006015                                  BNE     5$                      ;IF NOT, CONTINUE
006015  050000  006023                  ^O050000,5$
006017  104207  000056                  MOV     #56,R0                  ;SET '.'
006021                                  RETURN
006021  000000  000000                  ^O0,0
006023  105207  000022          5$:     ADD     #22,R0                  ;ELSE ADD 22(OCTAL)FOR NUMERAL CHARACTER
006025                                  RETURN
```

```
        006025  000000  000000                    ^00,0
```

```
RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE

                                        .SBTTL  RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
006027                                  RDSTAT:
                                        ;
                                        ;RETURN DRIVE STATUS
                                        ;STATUS RETURNED IN DM REGISTER 1
                                        ;
006027                                          PUSH    <R3,R0>                 ; SAVE R3 AND R0
006027  100463                                                                          MOV R3,-(SP)
006030  100467                                                                          MOV R0,-(SP)
006031  104203  000003                        MOV     #3,R3                   ; ALLOW ONLY 3 ERRORS
006033  060007                  STATLP: XFC     STATUS                  ; GET DRIVE'S STATUS
006034  103201  014000                        BIC     #14000,R1               ; CLEAR ERROR PASSING BITS
006036  102201  000400                        BIT     #XMTERR,R1              ; CHECK XMIT ERRORS
006040                                        BEQ     STATOK                  ; IF NO ERRORS, BRANCH
006040  010000  006051                        ^O10000,STATOK
006042  117403                                DEC     R3                      ; DECREMENT TRANSMIT ERROR COUNT
006043                                        BNE     STATLP                  ; IF ERROR COUNT INCOMPLETE, BRANCH
006043  050000  006033                        ^O50000,STATLP
006045  104201  010000                        MOV     #10000,R1               ; FLAG AS TRANSMIT ERROR
006047                                        BR      STATEX                  ; BRANCH
006047  000000  006062                        ^O0,STATEX
006051  102201  000004          STATOK: BIT     #RCVERR,R1              ; RECIEVER ERRORS
006053                                        BNE     STATEX                  ; IF VALID, BRANCH
006053  050000  006062                        ^O50000,STATEX
006055  117403                                DEC     R3                      ; DECREMENT ERROR COUNT
006056                                        BNE     STATLP                  ; IF ERROR COUNT NON-ZERO, BRANCH
006056  050000  006033                        ^O50000,STATLP
006060  104201  014000                        MOV     #14000,R1               ; FLAG AS INVALID STATUS ERROR
006062                          STATEX: POP     <R0,R3>                 ; RESTORE R0, R3
006062  104267                                                                          MOV (SP)+,R0
006063  104263                                                                          MOV (SP)+,R3
006064                                        RETURN                          ; RETURN TO CALLING MODULE
006064  000000  000000                        ^O0,0
```

```
                                .SBTTL  HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
006066                          HOSTRQ:
                                ;
                                ;SEND REQUEST BUFFER TO HOST AND WAIT FOR RESPONSE.
                                ;CLEAR ARGUMENT AREA OF OUT BUFFER IN PREPARATION
                                ;FOR NEXT HOSTRQ CALL.
                                ;
                                ;INPUTS:
                                ;       R0 - HOST REQUEST NUMBER
                                ;       OUT BUFFER LOADED WITH DATA

006066                                  PUSH    <R0,R1,R2>
006066  100467                                                                  MOV R0,-(SP)
006067  100461                                                                  MOV R1,-(SP)
006070  100462                                                                  MOV R2,-(SP)
006071  104070  002152                 MOV     R0,OUT.RQ               ; STORE REQUEST NUMBER IN BUFFER
006073  104207  002152          SNDAGN: MOV     #OUT.RQ,R0              ; SEND BUFFER TO HOST
006075  104201  000043                 MOV     #BUFSIZ,R1
006077  060016                         XFC     MRD
006100  115001                         TST     R1                      ; CHECK FOR ERROR
006101                                  BNE     SNDAGN                  ; IF ERROR, TRY AGAIN
006101  050000  006073                 ^050000,SNDAGN
006103  104207  002215                 MOV     #IN.RQ,R0               ; WAIT FOR RESPONSE FROM HOST
006105  104201  000043                 MOV     #BUFSIZ,R1
006107  060017                         XFC     MWR
006110  104200  177777  002152         MOV     #-1,OUT.RQ              ; MAKE REQUEST ILLEGAL
006113  104207  002153                 MOV     #OUT.01,R0              ; CLEAR ARGUMENT WORDS IN BUFFER
006115  104201  000042                 MOV     #BUFSIZ-1,R1            ; CLEAR ENTIRE BUFFER
006117  114002                         CLR     R2
006120  100272          CLRBUF: MOV     R2,(R0)+
006121  117401                         DEC     R1
006122                                  BPL     CLRBUF
006122  030000  006120                 ^030000,CLRBUF
006124                                  POP     <R2,R1,R0>
006124  104262                                                                  MOV (SP)+,R2
006125  104261                                                                  MOV (SP)+,R1
006126  104267                                                                  MOV (SP)+,R0
006127                                  RETURN
006127  000000  000000                 ^00,0
```

```
                                        ;
                                        ;       TALKER SENDS AND RECEIVES AN SDI INTERCHANGE
                                        ;
                                        ;INPUTS:
                                        ;       R2 - SDI INTERCONNECT
                                        ;       R3 - POINTER TO COMMAND TABLE CONTAINING APPROPRIATE COMMAND
                                        ;       SDILTO/SDISTO SDI LONG AND SHORT TIMEOUTS, RESPECTIVELY
                                        ;OUTPUTS:
                                        ;       R0 - RETURN OP CODE FROM UNIT
                                        ;       R3 - ERROR CODE - 0 = NO ERROR, 1 = RECEIVE ERROR, 100001 = SEND ERROR
                                        ;
 12 006131                              TALKER: PUSH    <R1,R4>         ; SAVE REGISTERS
    006131  100461                                                              MOV R1,-(SP)
    006132  100464                                                              MOV R4,-(SP)
 13 006133  104237                              MOV     (R3)+,R0        ; SET ADR OF SDI COMMAND BUFFER
 14 006134  104231                              MOV     (R3)+,R1        ; SET BUFFER LENGTH
 15 006135  060004                              XFC     SEND            ; SEND COMMAND
 16 006136  115001                              TST     R1              ; DID UNIT ACCEPT COMMAND
 17 006137                                      BEQ     TALK1A          ; IF SO, BRANCH
    006137  010000  006145                      ^010000,TALK1A
 18 006141  104203  100001                      MOV     #100001,R3      ; FLAG AS SEND ERROR
 19 006143                                      BR      TALK2B          ; BRANCH TO EXIT
    006143  000000  006224                      ^00,TALK2B
 20 006145  106203  002126              TALK1A: CMP     #LONG,R3        ; SEE IF LONG TIMEOUT TO BE USED
 21 006147                                      BMI     1$              ; IF SO, BRANCH
    006147  070000  006155                      ^070000,1$
 22 006151  104304  001625                      MOV     SDISTO,R4       ; SET UP SHORT TIMEOUT
 23 006153                                      BR      TALK1B          ; BRANCH
    006153  000000  006157                      ^00,TALK1B
 24 006155  104304  001626              1$:     MOV     SDILTO,R4       ; SET UP LONG TIMEOUT
 25 006157                              TALK1B:
 26 006157  115000  001622                      TST     SENDHR          ; DO WE SEND HOSTRQ?
 27 006161                                      BEQ     1$              ; IN NOT, CONTINUE
    006161  010000  006176                      ^010000,1$
 28 006163  104300  001623  002153            MOV     LUNIT,OUT.01    ; SEND UNIT NUMBER
 29 006166  114000  002154                      CLR     OUT.02          ; NO MEGABYTES READ
 30 006170  114000  002155                      CLR     OUT.03          ; NO MEGABYTES WRITTEN
 31 006172  104207  060011                      MOV     #T4MXFR,R0      ; SEND SOMETHING TO HOST
 32 006174                                      CALL    HOSTRQ          ;  SO HOST WON'T TIMEOUT DM PROG
    006174  020000  006066                      ^020000,HOSTRQ
 33 006176  104137                      1$:     MOV     (R3),R0         ; SET DATA BUFFER ADDRESS
 34 006177  104631  000001                      MOV     1(R3),R1        ; SET BUFFER LENGTH
 35 006201  060005                              XFC     RCV             ; SEND RECEIVE SDI COMMAND
 36 006202  115001                              TST     R1              ; DID ERROR OCCUR
 37 006203                                      BEQ     TALK2A          ; IF NOT, BRANCH
    006203  010000  006223                      ^010000,TALK2A
 38 006205  106201  000001                      CMP     #1,R1           ; SEE IF TIMEOUT
 39 006207                                      BEQ     2$              ; IF SO, BRANCH
    006207  010000  006214                      ^010000,2$
 40 006211  104013                              MOV     R1,R3           ; MOVE ERROR TYPE TO R3 FOR REPORTING
 41 006212                                      BR      TALK2B          ; EXIT
    006212  000000  006224                      ^00,TALK2B
 42 006214  117404                      2$:     DEC     R4              ; DECREMENT TIMEOUT VALUE
 43 006215                                      BNE     TALK1B          ; IF NOT TIMEOUT, BRANCH
    006215  050000  006157                      ^050000,TALK1B
 44 006217  104203  000001                      MOV     #1,R3           ; FLAG AS RECIEVE ERROR
 45 006221                                      BR      TALK2B          ; BRANCH TO EXIT
```

```
            006221  000000  006224                  ^O0,TALK2B
         46 006223  114003              TALK2A: CLR     R3          ; FLAG AS NO ERRORS
         47 006224                      TALK2B: POP     <R4,R1>     ; RESTORE R4, R1
            006224  104264                                                  MOV (SP)+,R4
            006225  104261                                                  MOV (SP)+,R1
         48 006226                              RETURN
            006226  000000  000000              ^O0,0
```

```
                                ;
                                ;XOR
                                ;
                                ;PERFORM XOR LOGIC FUNCTION ON TWO REGISTERS
                                ;INPUTS:
                                ;       R1, R2 - DATA TO BE XOR'ED
                                ;OUTPUTS:
                                ;       R1 - UNCHANGED
                                ;       R2 - XOR OF TWO INPUTS
                                ;
006230                          XOR:    PUSH R3                         ;SAVE R3
006230  100463                                                                  MOV R3,-(SP)
006231  104013                          MOV R1,R3
006232  103023                          BIC R2,R3
006233  103012                          BIC R1,R2
006234  101032                          BIS R3,R2
006235                          ;       POP R3                          ;RESTORE R3
006235  104263                                                                  MOV (SP)+,R3
006236  115002                          TST R2                          ;SET CONDITION CODES
006237                          ;       RETURN
006237  000000  000000                  ^O0,0
```

```
      1 006241                          TO:
      2                                 ;
      3                                 ;       CALCULATE THE TIMEOUT IN 9SEC INTERVALS (SDI RECEIVE XFC TAKES
      4                                 ;       9 SEC)
      5                                 ;
      6 006241  104201  000001                   MOV     #1,R1           ; SET UP LOG2 SHIFTER
      7 006243  110201                  1$:     ROL     R1              ; DOUBLE THE TIMEOUT VALUE
      8 006244  103201  000001                   BIC     #1,R1           ; CLEAR THE LOW BIT
      9 006246  117407                          DEC     R0              ; DECREMENT COUNT
     10 006247                                  BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
        006247  050000  006243                   ^O50000,1$
     11 006251  114007                          CLR     R0              ; CLEAR 9SEC COUNT
     12 006252  115407                  2$:     INC     R0              ; INCREMENT 9 SEC COUNT
     13 006253  107201  000011                   SUB     #9.,R1          ; SUBTRACT 9 SEC FROM TIMEOUT
     14 006255                                  BPL     2$              ; IF MORE TIME TO GO, BRANCH
        006255  030000  006252                   ^O30000,2$
     15 006257                                  RETURN                  ; RETURN TO CALLING PROGRAM
        006257  000000  000000                   ^O0,0
```

```
HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED,

                                        ;RERROR
                                        ;
                                        ;REPORT ERROR TO HOST PROGRAM
                                        ;THIS ROUTINE IS CALLED BY THE ERROR MACROS:
                                        ;ERRSF, ERRDF, ERRHRD AND ERRSFT
                                        ;
006261                                  RERROR: PUSH R0                 ;SAVE ONE REGISTER
006261  100467                                          MOV R0,-(SP)
006262  104067                                  MOV SP,R0               ;GET STACK POINTER
006263                                          PUSH <R1,R2,R3,R4>      ;SAVE MORE REGISTERS
006263  100461                                                          MOV R1,-(SP)
006264  100462                                                          MOV R2,-(SP)
006265  100463                                                          MOV R3,-(SP)
006266  100464                                                          MOV R4,-(SP)
006267  115407                                  INC R0                  ;CHANGE SAVED STACK POINTER TO POINT TO
                                                                        ; ADDRESS OF LOCATION AFTER CALL
006270  104271                                  MOV (R0)+,R1            ;GET RETURN PC
006271  104202  002153                          MOV #OUT.01,R2          ;GET ADDRESS OF WHERE TO PUT DATA
006273  117401                                  DEC R1                  ;REDUCE TO PC OF ERROR CALL
006274  100221                                  MOV R1,(R2)+            ;PUT PC IN OUT BUFFER
006275  115401                                  INC R1                  ;GET BACK TO RETURN PC
006276  104213                                  MOV (R1)+,R3            ;GET ERROR NUMBER AND TYPE
006277  100223                                  MOV R3,(R2)+            ;PUT IN BUFFER
006300  104303  001623                          MOV LUNIT,R3            ;PUT UNIT NUMBER IN BUFFER
006302  100223                                  MOV R3,(R2)+
006303  104113                                  MOV (R1),R3             ;GET MESSAGE POINTER
006304  103203  170000                          BIC #^C007777,R3        ;CLEAR OTHER BITS
006306  100223                                  MOV R3,(R2)+            ;PUT IN OUT BUFFER
006307  104214                                  MOV (R1)+,R4            ;GET COUNT OF PARAMETERS
                                                                        ;R1 IS NOW POINTING TO INSTRUCTION AFTER ERROR CALL
006310  110704                                  SWAB R4
006311  110604                                  ROR R4                  ;EXTRACT NUMBER OF PARAMETERS FROM ERROR MACRO
006312  110604                                  ROR R4
006313  110604                                  ROR R4
006314  110604                                  ROR R4
006315  103204  177760                          BIC #177760,R4
006317  104040  006343                          MOV R4,SPADJU+1         ;SAVE FOR LATER ADJUSTMENT OF STACK POINTER
006321                                          BEQ RERRCA              ;BRANCH IF NO PARAMETERS
006321  010000  006330                          ^O10000,RERRCA
006323  104273                          RERRPA: MOV (R0)+,R3            ;GET PARAMETER
006324  100223                                  MOV R3,(R2)+            ;STORE PARAMETER IN OUT BUFFER
006325  117404                                  DEC R4                  ;COUNT THE PARAMETERS
006326                                          BNE RERRPA              ;GET NEXT IF MORE
006326  050000  006323                          ^O50000,RERRPA
006330  100471                          RERRCA: MOV R1,-(R0)            ;PUT RETURN ADDRESS ON STACK
006331  104207  060013                          MOV #ERRMES,R0          ;SEND ERROR PACKET TO HOST PROGRAM
006333                                          CALL HOSTRQ
006333  020000  006066                          ^O20000,HOSTRQ
006335                                          POP <R4,R3,R2,R1,R0>    ;RESTORE REGISTERS
006335  104264                                                          MOV (SP)+,R4
006336  104263                                                          MOV (SP)+,R3
006337  104262                                                          MOV (SP)+,R2
006340  104261                                                          MOV (SP)+,R1
006341  104267                                                          MOV (SP)+,R0
006342  105206  000000                  SPADJU: ADD #0,SP               ;ADJUST STACK OVER PARAMETERS
                                                                        ;VALUE CHANGED ABOVE
006344                                          RETURN
```

HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED,

```
        006344  000000  000000                  ^00,0
```

HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED,

```
      1
```

```
                                        ;MESSAGE STORAGE OVERLAY

                                        FREE:                                                   ; FREE AREA STARTS HERE
                                                DMOVLY MS,0
 000346  106660                                 .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                                .NLIST BEX
 000000     042     110     117        MS2000: .ASCII\"HOST SPECIFIED UNIT #"D16" THAT CAN'T BE FOUND."N\
 000031     042     124     105                .ASCII\"TEST2 RESTARING"N\
 000042     000                                .BYTE   0
 000043     042     103     101        MS2001: .ASCII\"CANNOT RECEIVE VALID DRIVE STATE FROM DRIVE AFTER DRIVE WAS INITED"N\
 000105     042     103     110                .ASCII\"CHECK IF DRIVE IS POWERED ON."N\
 000125     000                                .BYTE 0
 000126     042     104     122        MS2002: .ASCII\"DRIVE STATE RECEIVED HAS BAD PARITY AFTER DRIVE WAS INITED"N\
 000164     000                                .BYTE 0
 000165     042     104     122        MS2003: .ASCII\"DRIVE IS NOT ASSERTING RECEIVER READY IN DRIVE STATE AFTER DRIVE WAS INITED
 000234     000                                .BYTE 0
 000235     042     124     111        MS2004: .ASCII\"TIME-OUT ON SEND OF ECHO COMMAND TO DRIVE"N\
 000263     042     040     040                .ASCII\"  ECHO DATA  "H8N\
 000274     000                                .BYTE 0
 000275     042     105     122        MS2005: .ASCII\"ERROR DURING RECEIVE OF ECHO RESPONSE FROM DRIVE"N\
 000326     042     040     040                .ASCII\"  ECHO DATA  "H8N\
 000337     000                                .BYTE 0
 000340     042     105     103        MS2006: .ASCII\"ECHO COMMAND RESPONDED WITH DIFFERENT DATA"N\
 000366     042     040     040                .ASCII\"  ECHO DATA SENT"S6H16N\
 000402     042     040     040                .ASCII\"  ECHO DATA RECEIVED  "H16N\
 000420     000                                .BYTE 0
 000421     042     105     122        MS2007: .ASCII\"ERROR BIT SET IN GET STATUS RESPONSE AFTER DRIVE CLEAR COMMAND"N\
 000461     042     040     040                .ASCII\"  GET STATUS RESPONSE  "NR1\
 000477     000                                .BYTE 0
 000500     042     124     111        MS2008: .ASCII\"TIME-OUT ON SEND OF ONLINE COMMAND TO DRIVE"N\
 000527     000                                .BYTE 0
 000530     042     105     122        MS2009: .ASCII\"ERROR DURING RECEIVE OF ONLINE RESPONSE FROM DRIVE"R1\
 000563     000                                .BYTE 0
 000564     042     117     116        MS2010: .ASCII\"ONLINE COMMAND WAS UNSUCCESSFUL"NR1\
 000606     000                                .BYTE 0
 000607     042     117     116        MS2011: .ASCII\"ONLINE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE"N\
 000642     042     040     040                .ASCII\"  EXPECTED RESPONSE  "H8N\
 000657     042     040     040                .ASCII\"  ACTUAL RESPONSE"S4H8N\
 000673     000                                .BYTE 0
 000674     042     124     111        MS2012: .ASCII\"TIME-OUT ON SEND OF GET UNIT CHARACTERISTICS COMMAND TO DRIVE"N\
 000734     000                                .BYTE   0
 000735     042     105     122        MS2013: .ASCII\"ERROR DURING RECEIVE OF GET UNIT CHARACTERISTICS COMMAND FROM DRIVE"R1\
 001000     000                                .BYTE   0
 001001     042     107     105        MS2014: .ASCII\"GET UNIT CHARACTERISTICS COMMAND WAS UNSUCCESSFUL"NR1\
 001034     000                                .BYTE   0
 001035     042     107     105        MS2015: .ASCII\"GET UNIT CHARACTERISTICS COMMAND DID NOT RETURN EXPECTED RESPONSE CODE"N\
 001101     042     040     040                .ASCII\"  EXPECTED RESPONSE  "H8N\
 001116     042     040     040                .ASCII\"  ACTUAL RESPONSE"S4H8N\
 001132     000                                .BYTE   0
 001133     042     040     040        MS2016: .ASCII\"  HOST PROGRAM GAVE DM CODE IMPROPER DATA"N\
 001161     042     040     040                .ASCII\"  EXPECTED VALUE SHOULD BE BETWEEN 0 AND 3"N\
 001207     042     040     040                .ASCII\"  ACTUAL VALUE WAS "D8N\
 001223     000                                .BYTE 0
 001224     042     124     111        MS2017: .ASCII\"TIME-OUT ON SEND OF DIAGNOSE COMMAND TO DRIVE"N\
 001254     000                                .BYTE 0
 001255     042     105     122        MS2018: .ASCII\"ERROR DURING RECEIVE OF DIAGNOSE RESPONSE FROM DRIVE"R1\
 001311     000                                .BYTE 0
```

```
     57 001312  042  104  111  MS2019: .ASCII\'DIAGNOSE COMMAND WAS UNSUCCESSFUL'NR1\
     58 001335  000                    .BYTE   0
     59 001336  042  104  111  MS2020: .ASCII\'DIAGNOSE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
     60 001372  042  040  040          .ASCII\''  EXPECTED RESPONSE  ''H8N\
     61 001407  042  040  040          .ASCII\''  ACTUAL RESPONSE''S4H8N\
     62 001423  000                    .BYTE 0
     63 001424  042  104  122  MS2021: .ASCII\'DRIVE DIAGNOSTIC REPORTS A HARD ERROR'N\
     64 001450  042  040  040          .ASCII\''  TEST NUMBER  ''H16N\
     65 001462  042  040  040          .ASCII\''  DRIVE TYPE   ''H8N\
     66 001474  042  040  040          .ASCII\''  ERROR NUMBER ''H16N\
     67 001507  122  062               .ASCII\R2\
     68 001510  000                    .BYTE 0
     69 001511  042  110  117  MS2022: .ASCII\'HOST PROGRAM DOWN LINE LOADED A DIAGNOSTIC WITH A ZERO BYTE COUNT'N\
     70 001553  000                    .BYTE 0
     71 001554  042  104  111  MS2023: .ASCII\'DIAGNOSTIC ''A6'' REQUESTED BY THE DRIVE COULD NOT BE SUPPLIED BY HOST.'N\
     72 001620  000                    .BYTE   0
     73 001621  042  124  111  MS2024: .ASCII\'TIME-OUT ON SEND OF MEMORY READ COMMAND TO DRIVE'N\
     74 001652  000                    .BYTE 0
     75 001653  042  105  122  MS2025: .ASCII\'ERROR DURING RECEIVE OF MEMORY READ PESPONSE FROM DRIVE'NR1\
     76 001711  000                    .BYTE 0
     77 001712  042  115  105  MS2026: .ASCII\'MEMORY READ COMMAND WAS UNSUCCESSFUL'NR1\
     78 001736  000                    .BYTE   0
     79 001737  042  115  105  MS2027: .ASCII\'MEMORY READ COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
     80 001775  042  040  040          .ASCII\''  EXPECTED RESPONSE  ''H8N\
     81 002012  042  040  040          .ASCII\''  ACTUAL RESPONSE''S4H8N\
     82 002026  000                    .BYTE 0
     83 002027  042  124  111  MS2028: .ASCII\'TIME-OUT ON SEND OF MEMORY WRITE COMMAND TO DRIVE'N\
     84 002061  000                    .BYTE 0
     85 002062  042  105  122  MS2029: .ASCII\'ERROR DURING RECEIVE OF MEMORY WRITE RESPONSE FROM DRIVE'R1\
     86 002120  000                    .BYTE 0
     87 002121  042  115  105  MS2030: .ASCII\'MEMORY WRITE COMMAND WAS UNSUCCESSFUL'NR1\
     88 002146  000                    .BYTE   0
     89 002147  042  115  105  MS2031: .ASCII\'MEMORY WRITE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
     90 002205  042  040  040          .ASCII\''  EXPECTED RESPONSE  ''H8N\
     91 002222  042  040  040          .ASCII\''  ACTUAL RESPONSE''S4H8N\
     92 002236  000                    .BYTE   0
     93 002237  042  124  111  MS2032: .ASCII\'TIME-OUT ON SEND OF RUN COMMAND TO DRIVE'N\
     94 002264  000                    .BYTE 0
     95 002265  042  105  122  MS2033: .ASCII\'ERROR DURING RECEIVE OF RUN RESPONSE FROM DRIVE'R1\
     96 002316  000                    .BYTE 0
     97 002317  042  122  125  MS2034: .ASCII\'RUN COMMAND WAS UNSUCCESSFUL'NR1\
     98 002337  000                    .BYTE 0
     99 002340  042  122  125  MS2035: .ASCII\'RUN COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
    100 002372  042  040  040          .ASCII\''  EXPECTED RESPONSE  ''H8N\
    101 002407  042  040  040          .ASCII\''  ACTUAL RESPONSE''S4H8N\
    102 002423  000                    .BYTE 0
    103 002424  042  124  111  MS2036: .ASCII\'TIME-OUT ON SEND OF RECALIBRATE COMMAND TO DRIVE'N\
    104 002455  000                    .BYTE 0
    105 002456  042  105  122  MS2037: .ASCII\'ERROR DURING RECEIVE OF RECALIBRATE RESPONSE FROM DRIVE'R1\
    106 002513  000                    .BYTE 0
    107 002514  042  122  105  MS2038: .ASCII\'RECALIBRATE COMMAND WAS UNSUCCESSFUL'NR1\
    108 002540  000                    .BYTE 0
    109 002541  042  122  105  MS2039: .ASCII\'RECALIBRATE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
    110 002577  042  040  040          .ASCII\''  EXPECTED RESPONSE  ''H8N\
    111 002614  042  040  040          .ASCII\''  ACTUAL RESPONSE''S4H8N\
    112 002630  000                    .BYTE 0
    113 002631  042  124  111  MS2040: .ASCII\'TIME-OUT ON SEND OF GET STATUS COMMAND TO DRIVE'N\
```

```
114 002662  000                                 .BYTE 0
115 002663  042  105  122  MS2041:  .ASCII\'ERROR DURING RECEIVE OF GET STATUS RESPONSE FROM DRIVE'R1\
116 002720  000                                 .BYTE 0
117 002721  042  107  105  MS2042:  .ASCII\'GET STATUS COMMAND WAS UNSUCCESSFUL'NR1\
118 002745  000                                 .BYTE 0
119 002746  042  107  105  MS2043:  .ASCII\'GET STATUS COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
120 003003  042  040  040           .ASCII\'  EXPECTED RESPONSE  '2H8N\
121 003020  042  040  040           .ASCII\'  ACTUAL RESPONSE'S4H8N\
122 003034  000                                 .BYTE 0
123 003035  042  124  111  MS2044:  .ASCII\'TIME-OUT ON SEND OF DRIVE CLEAR COMMAND TO DRIVE'N\
124 003066  000                                 .BYTE 0
125 003067  042  105  122  MS2045:  .ASCII\'ERROR DURING RECEIVE OF DRIVE CLEAR RESPONSE FROM DRIVE'R1\
126 003124  000                                 .BYTE 0
127 003125  042  104  122  MS2046:  .ASCII\'DRIVE CLEAR COMMAND WAS UNSUCCESSFUL'NR1\
128 003151  000                                 .BYTE 0
129 003152  042  104  122  MS2047:  .ASCII\'DRIVE CLEAR COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
130 003210  042  040  040           .ASCII\'  EXPECTED RESPONSE  '2H8N\
131 003225  042  040  040           .ASCII\'  ACTUAL RESPONSE'S4H8N\
132 003241  000                                 .BYTE 0
133 003242  101  064  116  MSG1:    .ASCII\A4N\
134 003243  042  111  116           .ASCII\'INFORMATION SENT BACK FROM THE DRIVE IS BEING PRESENTED.'N\
135 003301  042  040  040           .ASCII\'  TEST NUMBER  '2H16N\
136 003313  042  040  040           .ASCII\'  DRIVE TYPE   '2H8N\
137 003325  042  040  040           .ASCII\'  ERROR NUMBER '2H16N\
138 003340  122  062                .ASCII\R2\
139 003341  000                                 .BYTE 0
140 003342  042  106  117  MSG2:    .ASCII\'FOLLOWING REPORT HAS BEEN TRUNCATED DUE TO SIZE'N\
141 003373  000                                 .BYTE 0
142 003374  116  042  124  MSR0:    .ASCII\N'TIMEOUT ERROR OCCURED DURING RECEIVE XFC'N\
143 003422  000                                 .BYTE   0
144 003423  116  042  061  MSR1:    .ASCII\N'1ST WORD NOT START FRAME DURING RECEIVE XFC'N\
145 003452  000                                 .BYTE   0
146 003453  116                    MSR2:    .ASCII\N\
147 003453  042  106  122           .ASCII\'FRAMING ERROR OCCURED ON SDI LEVEL 0 READ DURING RECEIVE XFC'N\
148 003513  000                                 .BYTE   0
149 003514  116                    MSR3:    .ASCII\N\
150 003514  042  103  110           .ASCII\'CHECKSUM ERROR OCCURED ON SDI LEVEL 0 READ DURING RECEIVE XFC'N\
151 003554  000                                 .BYTE   0
152 003555  116                    MSR4:    .ASCII\N\
153 003555  042  102  125           .ASCII\'BUFFER SIZE SMALLER THEN RESPONSE DURING RECEIVE XFC'N\
154 003611  000                                 .BYTE   0
155 003612  116                    MSR5:    .ASCII\N\
156 003612  042  103  117           .ASCII\'CODE FROM RECEIVE XFC WAS UNINTELLIGIBLE FROM SUBSYSTEM  '2H16N\
157 003652  000                                 .BYTE   0
158 003653  042  125  116  ER5000:  .ASCII  \'UNABLE TO FIND REQUESTED DRIVE FOR TESTING'N\
159 003701  042  124  110           .ASCII  \'THE FOLLOWING IS VISIBLE ON THE PORTS'N\
160 003725  042  125  104           .ASCII  \'UDA PORT 0 -- '2R1\
161 003736  042  125  104           .ASCII  \'UDA PORT 1 -- '2R1\
162 003747  042  125  104           .ASCII  \'UDA PORT 2 -- '2R1\
163 003760  042  125  104           .ASCII  \'UDA PORT 3 -- '2R1\
164 003771  000                                 .BYTE   0
165 003772  042  116  117  SER10:   .ASCII  \'NO DRIVE ATTACHED'N\
166 004004  000                                 .BYTE   0
167 004005  042  122  103  SER11:   .ASCII  \'RCVR RDY NEVER ASSERTED'N\
168 004022  000                                 .BYTE   0
169 004023  042  124  111  SER12:   .ASCII  \'TIMEOUT OF SEND'N\
170 004034  000                                 .BYTE   0
```

```
    171 004035      042     124     111  SER13:  .ASCII  \"TIMEOUT OF RECEIVE'N\
    172 004047      000                          .BYTE   0
    173 004050      042     106     111  SER14:  .ASCII  \"FIRST WORD RECEIVED WAS NOT START FRAME'N\
    174 004075      000                          .BYTE   0
    175 004076      042     106     122  SER15:  .ASCII  \"FRAMING ERROR ON LEVEL 0 RECEIVE'N\
    176 004117      000                          .BYTE   0
    177 004120      042     103     110  SER16:  .ASCII  \"CHECKSUM ERROR ON LEVEL 0 RECEIVE'N\
    178 004142      000                          .BYTE   0
    179 004143      042     122     105  SER17:  .ASCII  \"RESPONSE LONGER THAN EXPECTED FOR CMD'N\
    180 004167      000                          .BYTE   0
    181 004170      042     104     122  SER18:  .ASCII  \"DRIVE "R1\
    182 004175      000                          .BYTE   0
    183 004176      104     061     062  SER18D: .ASCII  \D12",  "\       ; NOTE: THE STRING WITHIN THE \\'S <<MUST>>
    184 004202      104     061     062  SER18C: .ASCII  \D12",  "\       ; BE AN EVEN NUMBER OF CHAR, BECAUSE THE
    185 004206      104     061     062  SER18B: .ASCII  \D12",  "\       ; LABELS WILL FORCE WORD ALINGMENT, LEAVING
    186 004212      104     061     062  SER18A: .ASCII  \D12N\           ; JUNK IN THE LAST BYTE. (SER18A OK TO BE ODD)
    187 004214      000                          .BYTE   0
    188 004215      042     104     122  SER40:  .ASCII  \"DRIVE NOT AVAILABLE TO THIS UDA'N\
    189 004236      000                          .BYTE   0
    190 004237      042     125     116  SER41:  .ASCII  \"UNSPINABLE DRIVE "R1\
    191 004251      000                          .BYTE   0
    192 004252      042     122     105  STAT0:  .ASCII\"REAL TIME STATE"S2H16N\
    193 004265      042     123     124          .ASCII\"STATUS (R TO L)"S2H16S2H16S2H16S2H16S2H16S2H16S2H16N\
    194 004320      000                          .BYTE   0
    195 004320                                   DMEND
        004321  013364                           .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
    196         000001                           .END
```

SYMBOL TABLE

```
ARGS$ = 000003   DRVID = 000004   GETSTA= 000011   IN.25   002246   MS2021  001424
ATTN  = 000002   DRVONL= 000213   GETSUB= 000210   IN.26   002247     2022  001511
AVAIL = 000100   DRVRUN= 000014   GRPCYL= 000002   IN.27   002250   MS2023  001554
BF.DAT= 000000   DU.DFL= 020000   GRPOFF= 000011   IN.28   002251   MS2024  001621
BF.ECC= 000401   DU.FTL= 050000   GSTS3   003306   IN.29   002252   MS2025  001653
BF.EDC= 000400   DU.INF= 030000   GSTS6   003310   IN.30   002253   MS2026  001712
BREAK = 000000   DU.QUE= 010000   GSTS7   003421   IN.31   002254   MS2027  001737
BUFFLG= 040000   DU.SPC= 060000   GTSTAT  005242   IN.32   002255   MS2028  002027
BUFSIZ= 000043   DU.TER= 040000   GT.CMD  003575   IN.33   002256   MS2029  002062
BUF1    001673   D.LIMT= 000001   HBHINB= 007777   IN.34   002257   MS2030  002121
CHECK = 000010   D.SCHR= 000002   HBLONB= 170377   IRECLB= 000216   MS2031  002147
CHGMOD= 000201   ECC   = 000015   HDRPRE= 000005   LARGE = 000001   MS2032  002237
CHRRES= 000170   ECCFLG= 010000   HD.BAD= 110000   LBHINB= 177417   MS2033  002265
CLRBUF  006120   ECCRSH= 000002   HD.DBN= 140000   LBLONB= 177760   MS2034  002317
CLRDRV  005343   ECHO  = 000010   HD.LBN= 000000   LBNCYL= 000000   MS2035  002340
COMPAR= 000006   ECHOC = 000350   HD.PRV= 050000   LBNHST= 000012   MS2036  002424
COMPLT= 000176   ECHOD   001634   HD.RBN= 060000   LBNTRK= 000011   MS2037  002456
CONMEM  005472   ECHO1   003216   HD.REV= 030000   LINKLN= 000007   MS2038  002514
CR.DIA  002126   ECHO1A  003220   HD.XBN= 120000   LOBYTE= 000377   MS2039  002541
CR.DRC  001650   ECHO2   003221   HEADER= 000002   LONG    002126   MS2040  002631
CR.GCR  002111   ECHO3   003245   HIBYTE= 177400   LONGTO= 000001   MS2041  002663
CR.GST  001642   ECHO4   003254   HICYL = 000001   LOW   = 000000   MS2042  002721
CR.INR  002144   ECHO5   003272   HIDBN = 000003   LUNIT   001623   MS2043  002746
CR.ONL  002117   EOC   = 100000   HILBN = 000002   MAXSND= 001750   MS2044  003035
CR.RDM  002073   ERECOV= 000006   HIMEM = 037777   MEDTYP= 000006   MS2045  003067
CR.RUN  002136   ERHARD= 100000   HIRBN = 000003   MESSAG= 060015   MS2046  003125
CR.WRM  001657   ERLEV = 000002   HIXBN = 000002   MICREV= 000003   MS2047  003152
CVT   = 000020   ERRMC = 060014   HOSTRQ  006066   MRD   = 000016   MWR   = 000017
DATAVL= 040000   ERRMES= 060013   HRDREV= 000003   MSG1    003242   NAM.1   001616
DATPRE= 000005   ERRN  = 004000   ILOOP   003153   MSG2    003342   NAM.2   001617
DBNCYL= 000022   ERRTYP= 100000   INR     002151   MSR0    003374   NAM.3   001620
DIA     002133   ERSOFT= 140000   INSEEK= 000012   MSR1    003423   NUMPTR= 000003
DIAGDR  003775   ER.END  005334   IN.RQ   002215   MSR2    003453   OK.END  005340
DIAGNO  003765   ER5000  003653   IN.01   002216   MSR3    003514   ONL     002124
DIAGNS  003737   EXIT  = 000021   IN.02   002217   MSR4    003555   OUT.RQ  002152
DIAG.1  001615   EX.END  005341   IN.03   002220   MSR5    003612   OUT.01  002153
DIA.EN= 000374   FB.DAT= 000000   IN.04   002221   MS2000  000000   OUT.02  002154
DIA.OP= 000003   FB.EDC= 000400   IN.05   002222   MS2001  000043   OUT.03  002155
DINIT = 000011   FCTSIZ= 000010   IN.06   002223   MS2002  000126   OUT.04  002156
DISCON= 000204   FFFC  = 177774   IN.07   002224   MS2003  000165   OUT.05  002157
DIV50   005747   FFFD  = 177775   IN.08   002225   MS2004  000235   OUT.06  002160
DONE  = 060016   FFFE  = 177776   IN.09   002226   MS2005  000275   OUT.07  002161
DONECD  003140   FNDASC  005764   IN.10   002227   MS2006  000340   OUT.08  002162
DO.DCL  004221   FORMAT= 000001   IN.11   002230   MS2007  000421   OUT.09  002163
DO.DC4  004216   FRAME = 000004   IN.12   002231   MS2008  000500   OUT.10  002164
DO.DIX  004257   FREE    006346   IN.13   002232   MS2009  000530   OUT.11  002165
DO.DIO  004607   FSTOP = 100000   IN.14   002233   MS2010  000564   OUT.12  002166
DO.DI1  004317   FTLDEV= 040000   IN.15   002234   MS2011  000607   OUT.13  002167
DO.DI2  004350   FTLSYS= 000000   IN.16   002235   MS2012  000674   OUT.14  002170
DO.DI3  004515   FT.BUF= 000000   IN.17   002236   MS2013  000735   OUT.15  002171
DO.DI7  004522   FT.HI = 000001   IN.18   002237   MS2014  001001   OUT.16  002172
DO.DI8  004525   FT.LOW= 000002   IN.19   002240   MS2015  001035   OUT.17  002173
DRC     001655   GCC.EN= 000170   IN.20   002241   MS2016  001133   OUT.18  002174
DRCBYT  001656   GCR     002116   IN.21   002242   MS2017  001224   OUT.19  002175
DRCLR1  003304   GETBCN  005536   IN.22   002243   MS2018  001255   OUT.20  002176
DRTYPE= 000007   GETCHR= 000207   IN.23   002244   MS2019  001312   OUT.21  002177
DRVCLR= 000005   GETST   001647   IN.24   002245   MS2020  001336   OUT.22  002200
```

SYMBOL TABLE

```
OUT.23  002201          RDM     002100          SEND  = 000004          ST.DIA  003531          T2STRT  002760
OUT.24  002202          RDMFM   004642          SENDHR  001622          ST.DR = 000040          T4BB1 = 060005
OUT.25  002203          RDM.EN= 000162          SER10   003772          ST.ERR= 000002          T4BB2 = 060006
OUT.26  002204          RDM.OP= 000215          SER11   004005          ST.FE = 000200          T4MPRM= 060003
OUT.27  002205          RDSTAT  006027          SER12   004023          ST.FO = 002000          T4MXFR= 060011
OUT.28  002206          READ    003701          SER13   004035          ST.MOD= 000001          T4SEEK= 060010
OUT.29  002207          RECAL   005146          SER14   004050          ST.MSK= 000000          T4SOFT= 060007
OUT.30  002210          REGS$ = 177777          SER15   004076          ST.OA = 000200          T4UPRM= 060004
OUT.31  002211          REGU$ = 000001          SER16   004120          ST.PE = 000040          UDADM2= 001000 G
OUT.32  002212          RERRCA  006330          SER17   004143          ST.PS = 000002          UDA50 = 000000
OUT.33  002213          RERROR  006261          SER18   004170          ST.RE = 000100          UDA52 = 000001
OUT.34  002214          RERRPA  006323          SER18A  004212          ST.RR = 000100          UNITNB  001411
OVERFL= 000002          RETS  = 000001          SER18B  004206          ST.RTY= 000003          UNITS   001371
OVE.MN= 001364          REVECT= 000004          SER18C  004202          ST.RU = 000001          UNIT0 = 000001
OVE.MS= 000000          REVS  = 000007          SER18D  004176          ST.SR = 000020          UNIT1 = 000002
OVL.MN= 004763          RM    = 000004          SER18E  001630          ST.STA= 000001          UNIT2 = 000004
OVL.MS= 004322          RREAL = 013400          SER40   004215          ST.S7 = 000400          UNIT3 = 000010
OVL...= 011305          RSTOP = 100000          SER41   004237          ST.UNT= 000000          UNSSUC= 000175
OVS.MN= 001040          RUN     002143          SHRTTO= 000000          ST.WE = 000010          UREAD = 000013
OVS.MS= 013006          RUNDR   005052          SNDAGN  006073          SUB   = 000013          UTOTST= 060012
OV... = 011725          RWRDY = 100000          SPADJU  006342          TALKER  006131          UWRITE= 000014
PHYSA = 001000          RW.ANG= 000006          SS    = 000001          TALK1A  006145          WAITSI= 000012
PORT0   003056          RW.BUF= 000001          ST      001415          TALK1B  006157          WBUFLN= 000401
PORT2   003057          RW.CMD= 000004          STACK   002320          TALK2A  006223          WCONT = 040000
PORT3   003113          RW.HI = 000003          START   002321          TALK2B  006224          WREAL = 122400
PORT4   003123          RW.LOW= 000002          STATEX  006062          TEST    003146          WRITE   003633
PORT5   003133          RW.SDI= 000005          STATLP  006033          TESTEX  003773          WRM     001664
PRMS  = 000002          RW.STA= 000000          STATOK  006051          TESTX   003127          WRM.OP= 000017
PTYPE$= 000020          SAFWRD  001627          STATUS= 000007          TIMEOU= 000001          WRONG = 000002
RBNTRK= 000004          SAVREG  001624          STATO   004252          TO      006241          WRTMEM  004751
RBUFLN= 000415          SAVRID  001414          STOSTA  005437          TOOBIG= 000001          WSTOP = 140000
RCONT = 000000          SAVSTA  001413          STSIZE= 000200          TRKGRP= 000003          XBNCYL= 000021
RCTCPS= 000001          SBCRES= 000167          STSRES= 000366          TYPERR  005674          XFERRT= 000000
RCTCSZ= 000014          SDI     001412          ST.C  = 000002          TOO     003511          XMTERR= 000400
RCV   = 000005          SDILTO  001626          ST.CON= 000002          TIMSIZ= 060000          XOR     006230
RCVERR= 000004          SDISTO  001625          ST.DB = 001000          T2CMD = 060002          XREAD = 000002
RCVRDY= 000001          SDIVER= 000000          ST.DF = 000020          T2DLL = 060001          XWRITE= 000003

. ABS.  023652      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  16193 WORDS  ( 64 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  70 PAGES
B:UDAT2.052,B:UDAT2.L52/C=[30,30]DMAC52/M,B:UDAT2
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

MS2000	26-41	60-6#
MS2001	28-26	60-9#
MS2002	28-28	60-12#
MS2003	28-37	60-14#
MS2004	29-24	60-16#
MS2005	29-26	60-19#
MS2006	29-33	60-22#
MS2007	30-27	60-26#
MS2008	30-38	60-29#
MS2009	30-41	60-31#
MS2010	30-48	60-33#
MS2011	30-50	60-35#
MS2012	31-10	60-39#
MS2013	31-13	60-41#
MS2014	31-20	60-43#
MS2015	31-22	60-45#
MS2016	32-36	60-49#
MS2017	36-23	60-53#
MS2018	36-27	60-55#
MS2019	36-38	60-57#
MS2020	36-40	60-59#
MS2021	37-29	60-63#
MS2022	39-30	60-69#
MS2023	40-25	60-71#
MS2024	42-30	60-73#
MS2025	42-34	60-75#
MS2026	42-42	60-77#
MS2027	42-44	60-79#
MS2028	43-26	60-83#
MS2029	43-30	60-85#
MS2030	43-38	60-87#
MS2031	43-40	60-89#
MS2032	.4-9	60-93#
MS2033	44-12	60-95#
MS2034	44-19	60-97#
MS2035	44-21	60-99#
MS2036	45-9	60-103#
MS2037	45-12	60-105#
MS2038	45-19	60-107#
MS2039	45-21	60-109#
MS2040	46-18	60-113#
MS2041	46-21	60-115#
MS2042	46-29	60-117#
MS2043	46-31	60-119#
MS2044	47-18	60-123#
MS2045	47-21	60-125#
MS2046	47-29	60-127#
MS2047	47-31	60-129#
MSG1	37-33	60-133#
MSG2	50-53	60-140#
MSR0	51-22	60-142#
MSR1	51-26	60-144#
MSR2	51-30	60-146#
MSR3	51-34	60-149#
MSR4	51-38	60-152#
MSR5	51-40	60-155#

CROSS REFERENCE TABLE (CREF V04.00)

MWR	2-18#	54-21												
NAM.1	18-26#	40-18*	40-21*	40-25	40-25									
NAM.2	18-27#	40-9*	40-24*	40-25	40-25									
NAM.3	18-28#	40-11*	40-15*	40-25	40-25									
NUMPTR	25-38#	37-33#	50-53#											
OK.END	46-25	46-34#	47-25											
ONL	19-33	19-34#												
OUT.01	20-7#	25-38*	32-15*	33-24*	34-16*	37-33*	39-5	41-5*	50-46	50-53*	50-55	54-23	55-28*	58-13
OUT.02	20-8#	25-38*	32-16*	33-23*	34-17*	37-33*	41-8*	41-10*	50-53*	55-29*				
OUT.03	20-9#	25-5*	33-25*	34-21*	41-12*	55-30*								
OUT.04	20-10#	25-38*	33-26*	34-22*	41-13*									
OUT.05	20-11#	25-6	33-27*	34-23*	37-11	48-6								
OUT.06	20-12#	37-14												
OUT.07	20-13#													
OUT.08	20-14#	50-32	50-36	50-72*	50-75*	50-77*								
OUT.09	20-15#	37-24												
OUT.10	20-16#													
OUT.11	20-17#													
OUT.12	20-18#													
OUT.13	20-19#													
OUT.14	20-20#													
OUT.15	20-21#													
OUT.16	20-22#													
OUT.17	20-23#													
OUT.18	20-24#													
OUT.19	20-25#													
OUT.20	20-26#													
OUT.21	20-27#													
OUT.22	20-28#													
OUT.23	20-29#													
OUT.24	20-30#													
OUT.25	20-31#													
OUT.26	20-32#													
OUT.27	20-33#													
OUT.28	20-34#													
OUT.29	20-35#													
OUT.30	20-36#													
OUT.31	20-37#													
OUT.32	20-38#													
OUT.33	20-39#													
OUT.34	20-40#	50-50	50-59											
OUT.RQ	20-6#	25-38*	25-39	54-13*	54-14	54-22*								
OV...	1-67	1-67	1-67#	60-4	60-4	60-4	60-4#	60-195	60-195#					
OVE.MN	1-67#	60-4												
OVE.MS	60-4#	60-195												
OVERFL	3-75#													
OVL...	1-67	1-67#	60-4	60-4#	60-195	60-195#								
OVL.MN	1-67	1-67	60-4	60-4	60-4#									
OVL.MS	60-195	60-195	60-195#											
OVS.MN	1-67#													
OVS.MS	60-4#													
PHYSA	1-67	1-67	1-67	1-67	1-67#									
PORT0	26-17	26-42	27-10#											
PORT2	27-13#	27-42												
PORT3	27-29#	27-33												
PORT4	27-30	27-34#												

CROSS REFERENCE TABLE (CREF V04.00)

PORT5 27-18 27-20 27-22 27-40#

PRMS
26-41 26-41 26-41 26-41 26-41 26-41 26-41 26-41 26-41 26-41 26-41# 28-26 28-26 28-26
28-26 28-26 28-26 28-26 28-26 28-26 28-26 28-26# 28-28 28-28 28-28 28-28 28-28 28-28
28-28 28-28 28-28 28-28 28-28# 28-37 28-37 28-37 28-37 28-37 28-37 28-37 28-37 28-37
28-37 28-37# 29-24 29-24 29-24 29-24 29-24 29-24 29-24 29-24 29-24 29-24 29-24# 29-26
29-26 29-26 29-26 29-26 29-26 29-26 29-26 29-26 29-26 29-26# 29-33 29-33 29-33 29-33
29-33 29-33 29-33 29-33 29-33 29-33 29-33# 30-27 30-27 30-27 30-27 30-27 30-27 30-27
30-27 30-27 30-27 30-27# 30-38 30-38 30-38 30-38 30-38 30-38 30-38 30-38 30-38 30-38
30-38# 30-41 30-41 30-41 30-41 30-41 30-41 30-41 30-41 30-41 30-41 30-41# 30-48 30-48
30-48 30-48 30-48 30-48 30-48 30-48 30-48 30-48 30-48# 30-50 30-50 30-50 30-50 30-50
30-50 30-50 30-50 30-50 30-50 30-50# 31-10 31-10 31-10 31-10 31-10 31-10 31-10 31-10
31-10 31-10 31-10# 31-13 31-13 31-13 31-13 31-13 31-13 31-13 31-13 31-13 31-13 31-13#
31-20 31-20 31-20 31-20 31-20 31-20 31-20 31-20 31-20 31-20 31-20# 31-22 31-22 31-22
31-22 31-22 31-22 31-22 31-22 31-22 31-22 31-22# 32-36 32-36 32-36 32-36 32-36 32-36
32-36 32-36 32-36 32-36# 36-23 36-23 36-23 36-23 36-23 36-23 36-23 36-23 36-23 36-23
36-23 36-23# 36-27 36-27 36-27 36-27 36-27 36-27 36-27 36-27 36-27 36-27# 36-38
36-38 36-38 36-38 36-38 36-38 36-38 36-38 36-38 36-38 36-38# 36-40 36-40 36-40 36-40
36-40 36-40 36-40 36-40 36-40 36-40 36-40# 37-29 37-29 37-29 37-29 37-29 37-29 37-29
37-29 37-29 37-29 37-29# 39-30 39-30 39-30 39-30 39-30 39-30 39-30 39-30 39-30 39-30
39-30# 40-25 40-25 40-25 40-25 40-25 40-25 40-25 40-25 40-25 40-25 40-25# 42-30 42-30
42-30 42-30 42-30 42-30 42-30 42-30 42-30 42-30 42-30# 42-34 42-34 42-34 42-34 42-34
42-34 42-34 42-34 42-34 42-34# 42-42 42-42 42-42 42-42 42-42 42-42 42-42 42-42 42-42
42-42 42-42 42-42# 42-44 42-44 42-44 42-44 42-44 42-44 42-44 42-44 42-44 42-44 42-44#
43-26 43-26 43-26 43-26 43-26 43-26 43-26 43-26 43-26 43-26 43-26# 43-30 43-30 43-30
43-30 43-30 43-30 43-30 43-30 43-30 43-30# 43-38 43-38 43-38 43-38 43-38 43-38 43-38
43-38 43-38 43-38 43-38 43-38# 43-40 43-40 43-40 43-40 43-40 43-40 43-40 43-40 43-40
43-40 43-40# 44-9 44-9 44-9 44-9 44-9 44-9 44-9 44-9 44-9 ,4-9 44-9# 44-12
44-12 44-12 44-12 44-12 44-12 44-12 44-12 44-12 44-12 44-12# 44-19 44-19 44-19 44-19
44-19 44-19 44-19 44-19 44-19 44-19 44-19# 44-21 44-21 44-21 44-21 44-21 44-21 44-21
44-21 44-21 44-21 44-21# 45-9 45-9 45-9 45-9 45-9 45-9 45-9 45-9 45-9 45-9 45-9
45-9# 45-12 45-12 45-12 45-12 45-12 45-12 45-12 45-12 45-12 45-12 45-12# 45-19 45-19
45-19 45-19 45-19 45-19 45-19 45-19 45-19 45-19 45-19# 45-21 45-21 45-21 45-21 45-21
45-21 45-21 45-21 45-21 45-21 45-21# 46-18 46-18 46-18 46-18 46-18 46-18 46-18 46-18
46-18 46-18 46-18# 46-21 46-21 46-21 46-21 46-21 46-21 46-21 46-21 46-21 46-21 46-21#
46-29 46-29 46-29 46-29 46-29 46-29 46-29 46-29 46-29 46-29 46-29# 46-31 46-31 46-31
46-31 46-31 46-31 46-31 46-31 46-31 46-31# 47-18 47-18 47-18 47-18 47-18 47-18 47-18
47-18 47-18 47-18 47-18 47-18# 47-21 47-21 47-21 47-21 47-21 47-21 47-21 47-21 47-21
47-21 47-21# 47-29 47-29 47-29 47-29 47-29 47-29 47-29 47-29 47-29 47-29 47-29# 47-31
47-31 47-31 47-31 47-31 47-31 47-31 47-31 47-31 47-31 47-31 47-31#

PTYPE$
26-41 26-41 26-41# 29-24 29-24 29-24# 29-26 29-26 29-26# 29-33 29-33 29-33 29-33 29-33#
29-33# 30-41 30-41 30-41 30-41 30-41# 30-41# 30-50 30-50 30-50 30-50 30-50# 30-50# 31-13
31-13 31-13 31-13 31-13# 31-13# 31-22 31-22 31-22 31-22 31-22# 31-22# 32-36 32-36 32-36#
36-27 36-27 36-27 36-27 36-27# 36-27# 36-40 36-40 36-40 36-40 36-40# 36-40# 40-25 40-25
40-25 40-25 40-25 40-25# 40-25# 40-25# 42-34 42-34 42-34 42-34 42-34# 42-34# 42-44
42-44 42-44 42-44 42-44# 42-44# 43-30 43-30 43-30 43-30 43-30# 43-30# 43-40 43-40 43-40
43-40 43-40# 43-40# 44-12 44-12 44-12 44-12 44-12# 44-12# 44-21 44-21 44-21 44-21 44-21#
44-21# 45-12 45-12 45-12 45-12 45-12# 45-12# 45-21 45-21 45-21 45-21 45-21# 45-21# 46-21
46-21 46-21 46-21 46-21# 46-21# 46-31 46-31 46-31 46-31 46-31# 46-31# 46-21 46-21 46-21
47-21 47-21# 47-21# 47-31 47-31 47-31 47-31 47-31# 47-31#

RBNTRK 3-31#
RBUFLN 1-156#
RCONT 1-128#
RCTCPS 3-8#
RCTCSZ 3-40#
RCV 2-8# 22-57 55-35
RCVERR 4-23# 53-17

CROSS REFERENCE TABLE (CREF V04.00)

```
        42-23    43-19    48-9
SDILTO  18-38#   31-31*   55-24
SDISTO  18-37#   31-4*    31-27*   55-22
SDIVER   3-4#
SEND     2-7#    22-43    55-15
SENDHR  18-31#   36-12*   36-17*   37-3*    55-26*
SER10   22-20    60-165#
SER11   22-29    60-167#
SER12   22-49    60-169#
SER13   22-65    60-171#
SER14   22-71    60-173#
SER15   22-75    60-175#
SER16   22-79    60-177#
SER17   22-81    60-179#
SER18   25-18    60-181#
SER18A  18-41    60-186#
SER18B  18-42    60-185#
SER18C  18-43    60-184#
SER18D  18-44    60-183#
SER18E  18-41#   25-26
SER40   22-34    60-188#
SER41   25-16    60-190#
SHRTTO   3-3#    31-24
SNDAGN  54-14#   54-18
SPADJU  58-32*   58-42#
SS       3-9#
ST      18-21#   19-12    19-15    19-19    19-25    19-30    19-33    19-39    19-43    19-45    23-8     30-20    30-21    31-24
        31-28    34-18    37-4     38-17    38-29    38-32    38-33*   38-35*   39-10    39-12    39-14    39-31    39-42    39-48*
        39-49*   41-12    48-13    48-16    49-28    50-47    50-56
ST.C     2-30#
ST.CON   2-29#
ST.DB    2-47#
ST.DF    2-44#
ST.DIA  32-2#
ST.DR    2-37#    32-3     38-18
ST.ERR   2-28#
ST.FE    2-41#    30-22
ST.FO    2-46#
ST.MOD   2-27#
ST.MSK   2-25#
ST.OA    2-35#
ST.PE    2-43#    30-22
ST.PS    2-39#
ST.RE    2-42#
ST.RR    2-36#
ST.RTY   2-31#
ST.RU    2-40#    35-2
ST.S7    2-48#
ST.SR    2-38#    35-6
ST.STA   2-26#
ST.UNT   2-24#
ST.WE    2-45#    30-22
STACK   17-8      21-6#    27-37
START   17-18     22-6#
STATO   48-7      60-192#
STATEX  53-16     53-18    53-22#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
STATLP  53-9#    53-14    53-20
STATOK  53-12    53-17#
STATUS   2-10#   48-10    53-9
STOSTA  30-26    30-47    31-19    36-37    42-41    43-37    44-18    45-18    46-28    47-28    48-4#
STSIZE  18-20#   19-25    39-32    39-34
STSRES   4-45#   46-24    46-31    46-31
SUB      3-22#
T00     31-11    31-14    31-16    31-21    31-24#
T1MSIZ   4-3#
T2CMD    4-5#    32-24
T2DLL    4-4#    39-16
T2STRT  26-16#
T4BB1    4-8#
T4BB2    4-9#
T4MPRM   4-6#
T4MXFR   4-12#   55-31
T4SEEK   4-11#
T4SOFT   4-10#
T4UPRM   4-7#
TALK1A  55-17    55-20#
TALK1B  55-23    55-25#   55-43
TALK2A  55-37    55-46#
TALK2B  55-19    55-41    55-45    55-47#
TALKER  30-34    31-6     36-16    42-24    43-20    44-5     45-5     46-11    47-11    55-12#
TEST    27-35    28-11#
TESTEX  32-29    35-22#
TESTX   27-37#   35-22
TIMEOU   3-61#
TO      31-26    31-30    57-1#
TOOBIG   3-69#
TRKGRP   3-28#
TYPERR  30-40    31-12    36-26    42-33    43-29    44-11    45-11    46-20    47-20    51-16#
UDA50    1-32#    1-47     1-70     1-91
UDA52    1-33#    1-38     1-55     1-65     1-86     17-10    58-46
UDADM2   1-67#
UNIT0    3-46#   26-26    27-25
UNIT1    3-47#
UNIT2    3-48#
UNIT3    3-49#
UNITNB  18-11#   26-50*   27-23*   27-39    40-3*
UNITS   18-6#    22-14    24-7     24-19    24-26    25-7     25-36    26-24    27-13
UNSSUC   4-42#   30-45    31-17    36-35    42-39    43-35    44-16    45-16    46-26    47-26
UREAD    2-14#   39-35
UTOTST   4-13#   24-4
UWRITE   2-15#
WAITSI   2-13#
WBUFLN   1-155#   1-156
WCONT    1-126#
WREAL    1-130#
WRITE   32-31    33-10#
WRM     19-19    19-20#   32-6*    32-7*    32-8*    32-9*    32-10*   33-11*   33-12*   33-17*   33-18*   33-19*   39-4*    39-21*
        39-40*   39-41*   39-45*   39-50    39-58*   39-75*
WRM.OP  18-48#   19-20
WRONG    3-65#
WRTMEM  32-12    33-22    39-54    43-18#
WSTOP    1-125#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

UDAT2 DISK RESIDENT DMACR X04.01 19-AUG-82 13:10:12
TABLE OF CONTENTS

```
                                   .TITLE  UDAT2   DISK RESIDENT

                              ;
                              ;
                              ; COPYRIGHT (C)  1981
                              ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                              ;
                              ;
                              ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                              ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                              ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                              ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                              ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                              ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                              ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                              ;
                              ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                              ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                              ; EQUIPMENT CORPORATION.
                              ;
                              ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                              ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                              ;
                              ;
                              ;
                              ;
                              ;
                              ;
                              ;
                              ;
                              ;
        000001                UDA50=1
        000000                UDA52=0
                              ;THIS PROGRAM SHALL BE ASSEMBLED WITH THE PROGRAM DMACRO
                              ;USING A COMMAND LINE SIMILAR TO:
                              ;
                              ;  UDAT2.BIN,UDAT2/C=[1,2]DMACRO,UDAT2


   000000                          DMCODE  UDADM2,0,714,3,0,1000
                                   .SBTTL  UDA DM PROGRAM PARAMETERS

                                   .LIST MEB
        003720                     ERRN=2000.                        ;START ERROR NUMBERS AT 2000.
                              ;
                              ;    EQUATES
                              ;
                              ;    HIGHEST USABLE LOCATION OF UDA MEMORY + 1
                              ;
        007774                HIMEM   =       7774              ; HIGH MEMORY+1
        007774                OVSTRT  =       7774              ; OVERLAY ADDRESS LOCATION
                              ;
                              ;
                              ;    OFFSETS FOR FORMAT TRACK TABLE
                              ;
        000000                FT.BUF  =       0.                ;BUFFER POINTER OFFSET
        000001                FT.HI   =       1.                ;HI ORDER HEADER OFFSET
```

```
                                ;
                                ;
                                ;       OFFSETS FOR FORMAT TRACK BUFFER
                                ;
000000                          FB.DAT  =       0.              ;FIRST DATA WORD OFFSET
000400                          FB.EDC  =       256.            ;EDC WORD OFFSET
                                ;
                                ;
                                ;       OFFSETS FOR READ/WRITE I/O CHAIN TABLES
                                ;
000000                          RW.STAT =       0.              ;STATUS AND NEXT BUFFER POINTER OFFSET
000001                          RW.BUF  =       1.              ;POINTER TO DATA BUFFER
000002                          RW.LOW  =       2.              ;HI ORDER EXPECTED HEADER
000003                          RW.HI   =       3.              ;LOW ORDER EXPECTED HEADER
000004                          RW.CMD  =       4.              ;SDI COMMAND AND HEAD ADDRESS
000005                          RW.SDI  =       5.              ;DUMMY SDI CONTROL BLOCK POINTER
000006                          RW.ANG  =       6.              ;THETA FROM INDEX
                                ;
                                ;       CONSTANTS FOR READ AND WRITE XFC'S
                                ;
140000                          WSTOP   =       140000          ; LAST ENTRY IN CHAIN FOR WRITE
040000                          WCONT   =       40000           ; WRITE CONTINUE
100000                          RSTOP   =       100000          ; LAST ENTRY IN CHAIN FOR READ
000000                          RCONT   =       0               ; READ CONTINUE
100000                          FSTOP   =       100000          ; LAST ENTRY IN CHAIN FOR FORMAT
122400                          WREAL   =       122400          ; WRITE REAL TIME ECOMMAND
013400                          RREAL   =       13400           ; READ REAL TIME COMMAND
010000                          ECCFLG  =       10000           ; ECC ERROR IN BUFFER BIT
100000                          EOC     =       100000          ; END OF CHAIN
040000                          BUFFLG  =       40000           ; BUFFER FULL OR EMPTY FLAG
                                ;
                                ;
                                ;       HEADER CODES
                                ;
000000                          HD.LBN  =       000000          ;GOOD LBN
060000                          HD.RBN  =       060000          ;GOOD RBN, PERHAPS UNUSED
030000                          HD.REV  =       030000          ;REVECTORED LBN
110000                          HD.BAD  =       110000          ;BAD BLOCK
050000                          HD.PRV  =       050000          ;PRIMARY REVECTORED BLOCK
120000                          HD.XBN  =       120000          ;XBN BLOCK
140000                          HD.DBN  =       140000          ;DBN BLOCK
                                ;
                                ;       OFFSETS FOR DATA BUFFERS
                                ;
000000                          BF.DAT  =       0.              ;DATA
000400                          BF.EDC  =       256.            ;ERROR DETECTION CODE
000401                          BF.ECC  =       257.            ;LAST 17 ECC RESIDUES
                                ;
                                ;       BUFFER AND READ/WRITE CHAIN LINK SIZES
                                ;
000401                          WBUFLN  =       257.            ; WRITE BUFFER SIZE
000415                          RBUFLN  =       WBUFLN+12.      ; READ BUFFER SIZE
000007                          LINKLN  =       7.              ; LINK SIZE
```

```
                        ;       XFC DEFINITION EQUATES

000000          BREAK   =       0.              ;BREAKPOINT XFC CODE
000001          FORMAT  =       1.              ;FORMAT TRACK XFC CODE
000002          XREAD   =       2.              ;READ N SECTORS XFC CODE
000003          XWRITE  =       3.              ;WRITE N SECTORS XFC CODE
000004          SEND    =       4.              ;SEND SDI COMMAND XFC CODE
000005          RCV     =       5.              ;RECEIVE SDI MESSAGE XFC CODE
000006          COMPARE =       6.              ;COMPARE DATA PATTERN TO BUFFER
000007          STATUS  =       7.              ;RETURN DRIVE STATUS XFC CODE
000010          ECHO    =       8.              ;ECHO DATA TO DRIVE XFC CODE
000011          DINIT   =       9.              ;DRIVE INITIALIZE XFC CODE
000012          WAITSI  =       10.             ;WAIT FOR SECTOR OR INDEX PULSE
000013          UREAD   =       11.             ;READ UNIBUS MEMORY XFC CODE
000014          UWRITE  =       12.             ;WRITE UNIBUS MEMORY XFC CODE
000015          ECC     =       13.             ;DO ECC ON BUFFER XFC CODE
000016          MRD     =       14.             ;SEND BUFFER TO MAINTENANCE READ COMMAND
000017          MWR     =       15.             ;GET BUFFER FROM MAINTENANCE WRITE COMMAND
000020          CVT     =       16.             ;CONVERT TO PHYSICAL ADDRESS XFC CODE
000021          EXIT    =       17.             ;TERMINATE DM PROGRAM XFC CODE
                        ;
                        ;       GET STATUS OFFSETS
                        ;
000000          ST.UNT  =       0.              ;UNIT NUMBER
000000          ST.MSK  =       0.              ;SUBUNIT MASK
000001          ST.STA  =       1.              ;STATUS BYTE
000001          ST.MOD  =       1.              ;MODE BYTE
000002          ST.ERR  =       2.              ;ERROR BYTE
000002          ST.CON  =       2.              ;CONTROLLER BYTE
000002          ST.C    =       2.              ;C BITS
000003          ST.RTY  =       3.              ;RETRY COUNT/FAILURE CODE
                        ;
                        ;       STATUS BIT DEFINITIONS
                        ;
000200          ST.OA   =       200             ; ONLINE TO ANOTHER (SET IF DRIVE UNAVAILABLE)
000100          ST.RR   =       100             ; READJUSTMENT BIT (SET IF RECALIBRATION REQUIRED)
000040          ST.DR   =       40              ; DIAGNOSTIC REQUEST (SET IF DIAGNOSTIC REQUESTED)
000020          ST.SR   =       20              ; SPINDLE READY (SET IF SPINDLE READY)
000002          ST.PS   =       2               ; PORT SWITCH (SET IF PORT SWITCH IN)
000001          ST.RU   =       1               ; RUN/STOP SWITCH (SET IF RUN/STOP SWITCH IN)
000200          ST.FE   =       200             ; FATAL ERROR (SET IF FATAL ERROR OCCURRED)
000100          ST.RE   =       100             ; RETRIABLE ERROR (SET IF RETRIABLE ERROR OCCURRED)
000040          ST.PE   =       40              ; PROTOCOL ERROR (SET IF PROTOCOL ERROR OCCURRED)
000020          ST.DF   =       20              ; INITIALIZATION FAILURE (SET IF INIT FAILED)
000010          ST.WE   =       10              ; WRITE ENABLE (SET IF WRT ATTEMPTED ON PROT DISK)
002000          ST.FO   =       2000            ; FORMATTING (SET IF FORMATTING ENABLED)
001000          ST.DB   =       1000            ; DIAGNOSTIC CYLS (SET IF DIAG CYL ACCESS ENABLED)
000400          ST.S7   =       400             ; SECTOR SIZE (SET FOR 576 BYTE SECTORS)
```

```
                                ;       GET COMMON CHARACTERISTICS OFFSETS
                                ;
000000                          SHRTTO  =       0.              ;SHORT TIMEOUT <3:0>
000000                          SDIVER  =       0.              ;SDI VERSION <7:4>
000000                          XFERRT  =       0.              ;TRANSFER RATE <15:0>
000001                          LONGTO  =       1.              ;LONG TIMEOUT <3:0>
000001                          RETS    =       1.              ;RETRIES <7:4>
000001                          RCTCPS  =       1.              ;F/RCT COPIES <11:8>
000001                          SS      =       1.              ;SECTOR SIZE <15:15>
000002                          ERLEV   =       2.              ;ERROR RETRY LEVELS <7:0>
000002                          ECCRSH  =       2.              ;ECC THRESHOLD <15:8>
000003                          MICREV  =       3.              ;MICROCODE REVISION NUMBER <7:0>
000003                          HRDREV  =       3.              ;HARDWARE REVISION NUMBER <15:8>
000004                          DRVID   =       4.              ;UNIQUE DRIVE ID <47:0>
000007                          DRTYPE  =       7.              ;DRIVE TYPE IDENTIFIER <7:0>
000007                          REVS    =       7.              ;REVS/SECOND <15:8>
                                ;
                                ;       GET SUBUNIT CHARACTERISTICS OFFSETS
                                ;
                                ;THESE OFFSETS ARE CURRENTLY GIVEN AS FOLLOWING THE COMMON CHARACTERISTICS
                                ;
000013                          SUB     =       11.             ;OFFSET TO PUT SUBUNIT AFTER COMMON;
000000                          LBNCYL  =       0.              ;NUMBER OF CYLINDERS IN LBN AREA <31:0>
000001                          HICYL   =       1.              ;HI ORDER CYLINDER BITS <15:12>
000002                          GRPCYL  =       2.              ;GROUPS PER CYLINDER <7:0>
000002                          HILBN   =       2.              ;HI STARTING LBN <11:8>
000002                          HIXBN   =       2.              ;HI STARTING XBN <15:12>
000003                          TRKGRP  =       3.              ;TRACKS PER GROUP <7:0>
000003                          HIRBN   =       3.              ;HI STARTING RBN <11:8>
000003                          HIDBN   =       3.              ;HI STARTING DBN <15:12>
000004                          RBNTRK  =       4.              ;RBNS PER TRACK <6:0>
000004                          RM      =       4.              ;REMOVABLE MEDIA <7:7> 1=REMOVEABLE
000005                          DATPRE  =       5.              ;DATA PREAMBLE SIZE IN WORDS <7:0>
000005                          HDRPRE  =       5.              ;HEADER PREAMBLE SIZE IN WORDS <15:8>
000006                          MEDTYP  =       6.              ;MEDIA TYPE <31:0>
000010                          FCTSIZ  =       8.              ;FCT COPY SIZE <15:0>
000011                          LBNTRK  =       9.              ;LBNS PER TRACK <7:0>
000011                          GRPOFF  =       9.              ;GROUP OFFSET (SECTORS) <15:8>
000012                          LBNHST  =       10.             ;LBNS IN HOST AREA <31:0>
000014                          RCTCSZ  =       12.             ;RCT COPY SIZE <15:0>
000021                          XBNCYL  =       17.             ;CYLS IN XBN AREA <15:0>
000022                          DBNCYL  =       18.             ;CYLS IN DBN AREA <15:8>
                                ;
                                ;       UNIT CODES
                                ;
000001                          UNIT0   =       1.              ;UNIT ZERO CODE
000002                          UNIT1   =       2.              ;UNIT ONE CODE
000004                          UNIT2   =       4.              ;UNIT TWO CODE
000010                          UNIT3   =       8.              ;UNIT THREE CODE
                                ;
                                ;       BIT MASK DEFINITIONS
                                ;
                                ;
177400                          HIBYTE  =       177400          ;HIGH BYTE MASK
000377                          LOBYTE  =       000377          ;LOW BYTE MASK
007777                          HBHINB  =       7777            ;HI BYTE, HI NIBBLE MASK
170377                          HBLONB  =       170377          ;HI BYTE, LO NIBBLE MASK
```

```
     58          177417                LBHINB  =       177417          ;LO BYTE, HI NIBBLE MASK
     59          177760                LBLONB  =       177760          ;LO BYTE, LO NIBBLE MASK
     60                                ;
     61          000001                TIMEOUT =       1.              ;DRIVE TIMEOUT CODE
     62          000002                HEADER  =       2.              ;HEADER COMPARE FAILURE CODE
     63          000004                REVECT  =       4.              ;REVECTOR NEEDED CODE
     64                                ;
     65          000002                WRONG   =       2.              ;FIRST WORD NOT START FRAME CODE
     66          000004                FRAME   =       4.              ;FRAMING ERROR CODE
     67          000010                CHECK   =       8.              ;CHECKSUM ERROR CODE
     68                                ;
     69          000001                TOOBIG  =       1.              ;NUMBER OF WORDS EXCEEDS 7064
     70          000002                LOW     =       2.              ;DM BUFFER ADDRESS IS LESS THAN 7*4
     71                                ;
     72                                ;       BIT MASK
     73                                ;
     74          000001                LARGE   =       1.              ;BLOCK NUMBER TOO LARGE
     75          000002                OVERFL  =       2.              ;SECTOR NUMBER LARGER THAN 16 BITS
```

```
                                ;MAINTANENCE READ/WRITE REQUEST NUMBERS
                                ;
060000                          T1MSIZ  =       0.+DU.SPC       ;GET FREE MEMORY PARAMETERS
060001                          T2DLL   =       1.+DU.SPC       ;DOWNLINE LOAD DRIVE DIAGNOSTIC
060002                          T2CMD   =       2.+DU.SPC       ;MANUAL INTERVENTION TEST 2 PROTOCOL
060003                          T4MPRM  =       3.+DU.SPC       ;GET MASTER PARAMETERS FROM SW QUESTIONS
060004                          T4UPRM  =       4.+DU.SPC       ;GET UNIT PARAMETERS FROM HW QUESTIONS
060005                          T4BB1   =       5.+DU.SPC       ;GET BAD BLOCKS (1 THRU 14)
060006                          T4BB2   =       6.+DU.SPC       ;GET REST OF BAD BLOCKS (15 AND 16)
060007                          T4SOFT  =       7.+DU.SPC       ;ADD TO SOFT ERROR AND ECC COUNT
060010                          T4SEEK  =       8.+DU.SPC       ;ADD 1 TO SEEK COUNT
060011                          T4MXFR  =       9.+DU.SPC       ;ADD TO MEGABITS READ AND WRITTEN
060012                          UTOTST  =       10.+DU.SPC      ;GET UNITS TO TEST
060013                          ERRMES  =       11.+DU.SPC      ;PRINT ERROR MESSAGE
060014                          ERRMC   =       12.+DU.SPC      ;TEST 4 ERROR REPORTING
060015                          MESSAG  =       13.+DU.SPC      ;INFORMATION MESSAGE
060016                          DONE    =       14.+DU.SPC      ;MARK DM PROGRAM AS NO LONGER RUNNING
                                ;
                                ;       OTHER BIT DEFINITIIONS
                                ;
000001                          RCVRDY  =       1               ; RECIEVER READY 1 = READY
000002                          ATTN    =       2               ; ATTENTION BIT FOR RETURN DRIVE SIGNALS XFC
000004                          RCVERR  =       4               ; RECIEVER ERROR
000100                          AVAIL   =       100             ; AVAILABLE 1 = AVAILABLE
000400                          XMTERR  =       400             ; TRANSMIT ERROR
100000                          RWRDY   =       100000          ; IF SET, UDA IS ABLE TO READ AND/OR WRITE TO DRIVE
                                ;
                                ;       SDI COMMANDS AND RESPONSES
                                ;
000204                          DISCON  =       204             ; DISCONNECT DRIVE
000006                          ERECOV  =       6               ; ERROR RECOVERY
000201                          CHGMOD  =       201             ; CHANGE MODE
000213                          DRVONL  =       213             ; DRIVE ONLINE
000014                          DRVRUN  =       14              ; DRIVE RUN
000005                          DRVCLR  =       5               ; DRIVE CLEAR OPCODE
000207                          GETCHR  =       207             ; GET CHARACTERISTICS
000210                          GETSUB  =       210             ; GET SUBUNIT CHARACTERISTICS
000011                          GETSTA  =       11              ; GET STATUS
000216                          IRECLB  =       216             ; RECALIBRATE
000012                          INSEEK  =       12              ; INITIATE SEEK
000176                          COMPLT  =       176             ; SUCCESSFUL COMPLETION
000175                          UNSSUC  =       175             ; UNSUCCESSFUL COMPLETION
000170                          CHRRES  =       170             ; GET CHARACTERISTICS RESPONSE
000167                          SBCRES  =       167             ; GET SUBUNIT CHARACTERISTICS RESPONSE
000366                          STSRES  =       366             ; GET STATUS RESPONSE
000350                          ECHOC   =       350             ; DIAGNOSTIC ECHO COMMAND AND RESPONSE
                                ;
                                ;       ERROR CODES
                                ;
000000                          FTLSYS  =          0            ; SYSTEM FATAL ERROR
040000                          FTLDEV  =      40000            ; DEVICE FATAL
100000                          ERHARD  =     100000            ; HARD ERROR
140000                          ERSOFT  =     140000            ; SOFT ERROR
```

```
                                ;       DUMMY SDI CONTROL BLOCK OFFSETS
                                ;
000001                          D.LIMT  =       1                       ; DUMMY SDI SEARCH LIMIT
000002                          D.SCHR  =       2                       ; DUMMY POINTER TO SUBUNIT CHAR-5
                                ;
                                ;       DUP MESSAGE TYPES
010000                          DU.QUE = 10000
020000                          DU.DFL = 20000
030000                          DU.INF = 30000
040000                          DU.TER = 40000
050000                          DU.FTL = 50000
060000                          DU.SPC = 60000
                                .SBTTL  MACRO DEFINITIONS
                                ;
                                ;       MESSAGE CONTROL TABLE MACRO
                                ;
                                        .MACRO MSG CMDBUF,CMDSZ,RPLBUF,RPLSZ,SUCCOM
                                        .WORD CMDBUF                    ;ADDRESS OF COMMAND
                                        .WORD CMDSZ                     ;SIZE OF COMMAND IN BYTES
                                        .WORD RPLBUF                    ;ADDRESS OF REPLY
                                        .WORD RPLSZ                     ;SIZE OF REPLY IN WORDS
                                        .IF NB NUMBER
                                        .WORD SUCCOM                    ; SUCCESSFUL COMPLETION CODE
                                        .ENDC
                                        .ENDM
```

MACRO DEFINITIONS

```
      1                                 .MACRO  BCS     LAB..
      2
      3                                                         BCC     .+2
      4                                                         BR      LAB..
                                        .ENDM
```

MACRO DEFINITIONS

```
                        ;       PUSH REGISTER MACRO
                        ;
                                .MACRO PUSH R9
                                .IRP X,<R9>
                                
                                MOV X,-(SP)
                                .ENDR
                                .ENDM
                        ;
                        ;       POP REGISTER MACRO
                        ;
                                .MACRO POP R9
                                .IRP X,<R9>
                                
                                MOV (SP)+,X
                                .ENDR
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;ERROR MACROS
                        ;THESE MACROS ARE CALLED TO REPORT ERRORS TO THE HOST PROGRAM.
                        ;THE MACRO NAMES ARE : ERRSF, ERRDF, ERRHRD, ERRSFT. EACH RESULTS IN THE HOST
                        ;BEING REQUESTED TO REPORT THE ERROR.
                        ;ARGUMENTS:     1 (MS$) MESSAGE POINTER
                        ;               2 (P1$) PARAMETER #1
                        ;               3 (P2$) PARAMETER #2
                        ;               4 (P3$) PARAMETER #3
                        ;               5 (P4$) PARAMETER #4
                        ;               6 (P5$) PARAMETER #5
                        ;               7 (P6$) PARAMETER #6
                        ;               8 (P7$) PARAMETER #7
                        ;               9 (P8$) PARAMETER #8
                        ;
                        ;THE MESSAGE POINTER MUST POINT TO AN ADDRESS IN THE OVERLAY 'MS'' IMMEDIATELY
                        ;FOLLOWING THE MAIN CODE. ANY ADDRESS MODE MAY BE USED (E.G. #MS1, @R2).
                        ;THE ADDRESS MUST CONTAIN AN ASCII FORMAT STRING TO DETERMINE THE MESSAGE
                        ;TO PRINT.
                        ;THE PARAMETER ARGUMENTS ARE OPTIONAL. THEY SHOULD BE SUPPLIED ONLY WHEN
                        ;THERE IS DATA TO BE PASSED TO THE HOST THAT WILL BE USED IN PRINTING THE
                        ;MESSAGE. THESE PARAMETER ARGUMENTS ARE THE ADDRESS OF DATA TO BE PASSED
                        ;USING ANY ADDRESSING MODE DESIRED.
                        ;ALL REGISTERS ARE RETURNED UNCHANGED. IT SHOULD BE NOTED THAT ARGUMENTS
                        ;CONTAINING SOMETHING OTHER THAN A REGISTER NAME (E.G. #100 OR MEMADR)
                        ;ASSEMBLE TO INSTRUCTIONS THAT SAVE AND RESTORE A REGISTER ON THE STACK.

                                .MACRO ERRSF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ FTLSYS,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRDF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ FTLDEV,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRHRD MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NLIST
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ ERHARD,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .LIST
                                .ENDM

                                .MACRO ERRSFT MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ ERSOFT,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM
```

```
                        ;THE FOLLOWING MACRO ACTUALLY PROCESSES THE ERROR CALL TO THE HOST PROGRAM

                                .MACRO ERROR$ ET$,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,ERRN$
                                  .RADIX 8
                                  PRMS=ARGS$-1
                                  .IF LT,<PRMS>
                                    .ERROR;NOT ENOUGH ARGUMENTS IN ERROR CALL
                                  .ENDC
                                  REGS$=-1
                                  .IIF GE,<PRMS-8.>,PARGS. P8$
                                  .IIF GE,<PRMS-7.>,PARGS. P7$
                                  .IIF GE,<PRMS-6.>,PARGS. P6$
                                  .IIF GE,<PRMS-5.>,PARGS. P5$
                                  .IIF GE,<PRMS-4.>,PARGS. P4$
                                  .IIF GE,<PRMS-3.>,PARGS. P3$
                                  .IIF GE,<PRMS-2.>,PARGS. P2$
                                  .IIF GE,<PRMS-1.>,PARGS. P1$
                                  .IF GE REGS$
                                    RSTR$ \REGS$
                                  .ENDC
                                  .RADIX 10
                                  .LIST
                                                                                CALL RERROR    ;ERROR # ERRN$'.
                                  .NLIST
                                  .RADIX 8
                                  .LIST
                                                                                .WORD ET$+ERRN
                                                                                .WORD <PRMS*10000>+MS$
                                  .NLIST
                                  ERRN=ERRN+1
                                .ENDM

                                .MACRO PARGS.,ADDR$
                                  .NTYPE PTYPE$,ADDR$
                                  .IF EQ,<PTYPE$&70>
                                    .IIF EQ,<PTYPE$&7>-REGS$,RSTR$ \REGS$
                                    .LIST
                                                                                MOV ADDR$,-(SP)
                                    .NLIST
                                  .IFF
                                    .IF EQ,<PTYPE$&7>-1          ;PICK A REGISTER TO USE
                                      REGU$=2                    ;SELECT R2 IF R1 IS USED IN PARAMETER FETCH
                                    .IFF
                                      REGU$=1                    ;OTHERWISE USE R1
                                    .ENDC
                                    .IF NE,<REGU$-REGS$>         ;IF REGISTER NOT ALREADY SAVED
                                      .IF GE,REGS$
                                        RSTR$ \REGS$             ;RESTORE CURRENT SAVED REGISTER
                                      .ENDC
                                      SAVR$ \REGU$               ;THEN SAVE SELECTED REGISTER
                                    .ENDC
                                    GETP$ \REGS$,ADDR$
                                  .ENDC
                                .ENDM

                                .MACRO  DEVFTL  NUM,ARGS
                                ERROR   FTLDEV,NUM,<ARGS>
```

```
                                                                        MOV     #ERRMC,OUT.RQ
                        .ENDM
                        .MACRO  ERROR   TYPE,NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       5
                        MOVMSG  #ER'NUM,4
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC
                                                                        MOV     #NUM!TYPE,R2
                                                                        MOV     R2,OUT.02
                                                                        MOV     #.,OUT.01
                        .RADIX
                        .ENDM
```

```
                        .MACRO SAVR$ REGN
                          .LIST
                                                        MOV R'REGN,SAVREG
                          .NLIST
                          REGS$=REGN
                        .ENDM

                        .MACRO RSTR$ REGN
                          .LIST
                                                        MOV SAVREG,R'REGN
                          .NLIST
                          REGS$=-1
                        .ENDM

                        .MACRO GETP$ REGN,ADDR$
                          .LIST
                                                        MOV ADDR$,R'REGN
                                                        MOV R'REGN,-(SP)
                          .NLIST
                        .ENDM
```

MACRO DEFINITIONS

```
 1                                      .MACRO  MSSGE   NUM,ARGS
 2                                      .RADIX  10
 3                                      NUMPTR  =       3
 4                                                                      MOV     LUNIT,OUT.01
 5                                      MOVMSG  #'NUM,2
 6                                      .IF     NB,<ARGS>
 7                                      .IRP    X,<ARGS>
 8                                      MOVMSG  X,\NUMPTR
 9                                      NUMPTR  =       NUMPTR + 1
10                                      .ENDR
11                                      .ENDC
12                                                                      PUSH    <R0,R1>
13                                                                      MOV     #MESSAG,R0
14                                                                      CALL    HOSTRQ
15                                                                      POP     <R1,R0>
16                                      .RADIX
17                                      .ENDM
18
19                                      .MACRO  MOVMSG  ARG,INDX
20
21                                      .IF     LT,INDX-10
22                                                                      MOV     ARG,OUT.0'INDX
23                                      .IFF
24                                                                      MOV     ARG,OUT.'INDX
25                                      .ENDC
26                                      .ENDM
```

MACRO DEFINITIONS

```
      1                                 ;ASSUME MACRO
      2                                         .MACRO  ASSUME  P1,P2
      3                                         .IF     NE,P1-P2
      4                                         .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
      5                                         .ENDC
      6                                         .ENDM
```

MACRO DEFINITIONS

```
                        ;RETURN DRIVE STATUS MACRO WITH ERROR REPORTING

                                .MACRO  DSTAT,LAB$,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   ME
                                .LIST
                                CALL    RDSTAT                  ; GET DRIVE STATUS
                                BIT     #10000,R1               ; SEE IF ANY ERRORS
                                BEQ     2$                      ; IF NO ERROR, BRANCH
                                BIT     #4000,R1                ; SEE IF XMIT ERROR
                                BEQ     1$                      ; IF SO, BRANCH
                                ERRHRD  E1                      ; REPORT INVALID STATUS ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        1$:     ERRHRD  E2                      ; REPORT XMIT ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        2$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

```
      1                                ;
      2                                ;       XOR THE CONTENTS OF TWO REGISTERS
      3                                ;
      4                                        .MACRO  RXOR    REG1,REG2
      5                                        MOV     REG2,-(SP)              ; SAVE REGISTER REG2
      6                                        BIC     REG1,REG2               ; CLEAR COORESPONDING BITS IN REG2
      7                                        BIC     (SP)+,REG1              ; CLEAR COORESPONDING BITS IN REG1
      8                                        BIS     REG1,REG2               ; OR WHAT'S LEFT
      9                                        .ENDM
```

```
      1                                ;
      2                                ;       CONVERT .BLKW CALLS TO ACTUAL WORDS GENERATED
      3                                ;
      4                                        .MACRO  .BLKW   COUNT
      5                                        .NLIST
      6                                        .REPT   COUNT
      7                                        .WORD   0
      8                                        .ENDR
      9                                        .LIST
     10                                        .ENDM
```

```
                        ;       SDI INTERCHANGE WITH DRIVE WITH ERROR REPORTING
                        ;
                                .MACRO  TALKX   ERRLAB,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   ME
                                .LIST
                                CALL    TALKER                  ; INITIATE SDI INTERCHANGE
                                TST     R3                      ; SEE IF ERROR OCCURRED
                                BEQ     12$                     ; IF NOT, BRANCH
                                BPL     11$                     ; IF SO, BRANCH
                                ERRHRD  E1;SEND COMMAND ERROR
                                BR      ERRLAB
                        11$:    ERRHRD  E2;RECEIVE COMMAND ERROR
                                BR      ERRLAB
                        12$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

START OF TEST CODE

```
      1                                 .SBTTL  START OF TEST CODE
      2                                 ;THE FOLLOWING IS FOR DEBUG PURPOSES ONLY. CAN BE NOP OR BREAKPOINT.
      3
      4 000714  114007                          CLR R0                          ;CHANGE TO BREAKPOINT FOR DEBUG
      5
      6                                 ;INITIALIZE STACK
      7
      8 000715  104206  001647                  MOV #STACK,SP                   ;SET UP STACK POINTER
     18 000717  001650                          BR      START                   ; BRANCH OVER SUPPORT CODE
```

PROGRAM VARIABLES

```
                                        .SBTTL  PROGRAM VARIABLES
                                ;PROGRAM VARIABLES

                                ;UNIT NUMBER STORAGE FOR DISK DRIVES TO TEST

000720  177777  177777  177777  UNITS:  .WORD   -1,-1,-1,-1           ;LOGICAL UNIT NUMBER IF POSITIVE
000723  177777
000724  177777  177777  177777          .WORD   -1,-1,-1,-1           ;DISK DRIVE NOT TO BE TESTED IF NEGATIVE
000727  177777
000730  177777  177777  177777          .WORD   -1,-1,-1,-1
000733  177777
000734  177777  177777  177777          .WORD   -1,-1,-1,-1
000737  177777

000740  000000          UNITNB: .WORD   0                             ;NUMBER OF UNIT CURRENTLY UNDER TEST
                                                                      ;POINTER INTO TABLE ABOVE

000741  000000          SDI:    .WORD   0                             ;SDI INTERCONNECT CODE FOR XFC CALLS

000742  000000          SAVSTA: .WORD   0                             ;SAVE STATUS
000743  000000          SAVRID: .WORD   0                             ;SAVE REGION ID

                                ;DRIVE RESPONSE BUFFERS
        000200          STSIZE  =       200
000744                  ST:     .BLKW   200

                                ;
001144  000000          DIAG.1: .WORD   0                             ; = 0, 1ST DIAGNOSE COMMAND
                                                                      ; NOT = 0, DRIVE TESTED BEFORE
001145  000000          NAM.1:  .WORD   0                             ;HOLD PROGRAM NAME
001146  000000          NAM.2:  .WORD   0
001147  000000          NAM.3:  .WORD   0
001150  000000                  .WORD   0

                                ;
001151  000000          SENDHR: .WORD   0                             ;IF 0, DON'T SEND HOSTRQ IN TALKER
                                                                      ;IF NOT 0, SEND HOSTRQ -> DOING DIAGNOSE COMMAND

                                ;
001152  177777          LUNIT:  .WORD -1                              ;LOGICAL UNIT NUMBER (-1 FOR NOT AVAILABLE)

001153  000000          SAVREG: .WORD   0                             ;STORAGE FOR REGISTER AT CALL TIME
001154  000012          SDISTO: .WORD   10.                           ; SDI SHORT TIMEOUT
001155  000024          SDILTO: .WORD   20.                           ; SDI LONG TIMEOUT
001156  000000          SAFWRD: .WORD   0                             ; SAFE WORD TO USE IN DO.DCL

001157  004212          SER18E: .WORD   SER18A                        ; FOR MULTIPLE SUBUNIT ERROR REPORTING
001160  004206                  .WORD   SER18B
001161  004202                  .WORD   SER18C
001162  004176                  .WORD   SER18D

        001750          MAXSND  =       1000.                         ; MAXIMUM # OF SENDS

        000017          WRM.OP  =       17                            ; WRITE MEMORY COMMAND
        000215          RDM.OP  =       215                           ; READ MEMORY COMMAND
        000003          DIA.OP  =       3                             ; DIAGNOSE COMMAND
        000207          GETCHR  =       207                           ; GET COMMON CHARACTERISTICS COMMAND
        000162          RDM.EN  =       162                           ; READ MEMORY RESPONSE
        000374          DIA.EN  =       374                           ; DIAGNOSE RESPONSE
```

```
     54          000170                  GCC.EN  =       170                     ; GET COMMON CHARACTERISTICS
     55                                  ;
     56                                  ;       DIAGNOSE ET AND DA
     57                                  ;
     58          100000                  ERRTYP  =       100000                  ;ERROR TYPE ET
     59          040000                  DATAVL  =        40000                  ;DATA AVAILABLE
     60                                  ;
     61                                  ;       REGION ID'S
     62                                  ;
     63          177774                  FFFC    =       FFFD - 1
     64          177775                  FFFD    =       177775
     65          177776                  FFFE    =       FFFD + 1
     66
```

```
                                                .SBTTL  SDI COMMANDS USED FOR TEST 2
001163     377     350          ECHOD:  .BYTE   377,ECHOC                ;ECHO DATA TO SEND TO DRIVE
001164     000     350                  .BYTE   000,ECHOC
001165     252     350                  .BYTE   252,ECHOC
001166     360     350                  .BYTE   360,ECHOC
001167     017     350                  .BYTE   017,ECHOC
001170  000000                          .WORD   0                          ;END MARKER

                                                .SBTTL  COMMAND BUFFERS FOR SEND XFC
                                        ;COMMAND BUFFERS FOR SEND XFC

001171                                  CR.GST: MSG     GETST,1,ST,7               ; GET STATUS SDI INFORMATION
001171  001176                                  .WORD GETST                        ;ADDRESS OF COMMAND
001172  000001                                  .WORD 1                    ;SIZE OF COMMAND IN BYTES
001173  000744                                  .WORD ST                           ;ADDRESS OF REPLY
001174  000007                                  .WORD 7                    ;SIZE OF REPLY IN WORDS
001175  000000                                  .WORD                      ; SUCCESSFUL COMPLETION CODE
001176     000     011          GETST:  .BYTE   0,GETSTA
                                        ;
001177                                  CR.DRC: MSG     DRC,2,ST,7                 ; DRIVE CLEAR SDI INFORMATION
001177  001204                                  .WORD DRC                          ;ADDRESS OF COMMAND
001200  000002                                  .WORD 2                    ;SIZE OF COMMAND IN BYTES
001201  000744                                  .WORD ST                           ;ADDRESS OF REPLY
001202  000007                                  .WORD 7                    ;SIZE OF REPLY IN WORDS
001203  000000                                  .WORD                      ; SUCCESSFUL COMPLETION CODE
001204     000     005          DRC:    .BYTE   0,DRVCLR
001205  177777                  DRCBYT: .WORD   177777
                                        ;
001206                                  CR.WRM: MSG     WRM,7,ST,7                 ; MEMORY WRITE COMMAND INFO
001206  001213                                  .WORD WRM                          ;ADDRESS OF COMMAND
001207  000007                                  .WORD 7                    ;SIZE OF COMMAND IN BYTES
001210  000744                                  .WORD ST                           ;ADDRESS OF REPLY
001211  000007                                  .WORD 7                    ;SIZE OF REPLY IN WORDS
001212  000000                                  .WORD                      ; SUCCESSFUL COMPLETION CODE
001213     000      017         WRM:    .BYTE   0,WRM.OP
001214  000000  000000  000000          .WORD   0,0,0
001217  000000  000000  000000          .WORD   0,0,0
001222                          BUF1:   .BLKW   200
                                        ;
001422                                  CR.RDM: MSG     RDM,6,ST,STSIZE            ; MEMORY READ COMMAND INFO
001422  001427                                  .WORD RDM                          ;ADDRESS OF COMMAND
001423  000006                                  .WORD 6                    ;SIZE OF COMMAND IN BYTES
001424  000744                                  .WORD ST                           ;ADDRESS OF REPLY
001425  000200                                  .WORD STSIZE                       ;SIZE OF REPLY IN WORDS
001426  000000                                  .WORD                      ; SUCCESSFUL COMPLETION CODE
001427     000      215         RDM:    .BYTE   0,RDM.OP
001430  000000  000000  000000          .WORD   0,0,0,0
001433  000000
001434  000000  000000  000000          .WORD   0,0,0,0
001437  000000

                                        ;
001440                                  CR.GCR: MSG     GCR,1,ST,12.               ; GET CHARACTERISTICS
001440  001445                                  .WORD GCR                          ;ADDRESS OF COMMAND
001441  000001                                  .WORD 1                    ;SIZE OF COMMAND IN BYTES
001442  000744                                  .WORD ST                           ;ADDRESS OF REPLY
001443  000014                                  .WORD 12.                          ;SIZE OF REPLY IN WORDS
001444  000000                                  .WORD                      ; SUCCESSFUL COMPLETION CODE
```

COMMAND BUFFERS FOR SEND XFC

```
     31 001445     000     207          GCR:    .BYTE   0,GETCHR                ; GET CHARACTERISTICS
     32                                 ;
     33 001446                          CR.ONL: MSG     ONL,2,ST,7              ; ONLINE COMMAND
        001446  001453                          .WORD ONL                       ;ADDRESS OF COMMAND
        001447  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
        001450  000744                          .WORD ST                        ;ADDRESS OF REPLY
        001451  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
        001452  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
     34 001453     000     213          ONL:    .BYTE   0,213
     35 001454  000077                          .WORD   77
     36                                 ;
     37 001455                          LONG:                                   ; ALL LONG TIMEOUT COMMANDS FOLLOW
     38                                 ;
     39 001455                          CR.DIA: MSG     DIA,3,ST,7              ; DIAGNOSE COMMAND INFORMATION
        001455  001462                          .WORD DIA                       ;ADDRESS OF COMMAND
        001456  000003                          .WORD 3                 ;SIZE OF COMMAND IN BYTES
        001457  000744                          .WORD ST                        ;ADDRESS OF REPLY
        001460  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
        001461  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
     40 001462     000     003          DIA:    .BYTE   0,DIA.OP
     41 001463  000000                          .WORD   0
     42 001464  000000                          .WORD   0
     43 001465                          CR.RUN: MSG     RUN,1,ST,7
        001465  001472                          .WORD RUN                       ;ADDRESS OF COMMAND
        001466  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
        001467  000744                          .WORD ST                        ;ADDRESS OF REPLY
        001470  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
        001471  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
     44 001472     000     014          RUN:    .BYTE   0,DRVRUN
     45 001473                          CR.INR: MSG     INR,1,ST,7
        001473  001500                          .WORD INR                       ;ADDRESS OF COMMAND
        001474  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
        001475  000744                          .WORD ST                        ;ADDRESS OF REPLY
        001476  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
        001477  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
     46 001500     000     216          INR:    .BYTE   0,IRECLB
```

```
                                .SBTTL  STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS
                        ;STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS

                        ;OUT BUFFER - DATA TO SEND TO HOST

001501  000000          OUT.RQ: .WORD 0                 ;HOST REQUEST CODE
001502  000000          OUT.01: .WORD 0                 ;DATA ARGUMENT  1
001503  000000          OUT.02: .WORD 0                 ;DATA ARGUMENT  2
001504  000000          OUT.03: .WORD 0                 ;DATA ARGUMENT  3
001505  000000          OUT.04: .WORD 0                 ;DATA ARGUMENT  4
001506  000000          OUT.05: .WORD 0                 ;DATA ARGUMENT  5
001507  000000          OUT.06: .WORD 0                 ;DATA ARGUMENT  6
001510  000000          OUT.07: .WORD 0                 ;DATA ARGUMENT  7
001511  000000          OUT.08: .WORD 0                 ;DATA ARGUMENT  8
001512  000000          OUT.09: .WORD 0                 ;DATA ARGUMENT  9
001513  000000          OUT.10: .WORD 0                 ;DATA ARGUMENT 10
001514  000000          OUT.11: .WORD 0                 ;DATA ARGUMENT 11
001515  000000          OUT.12: .WORD 0                 ;DATA ARGUMENT 12
001516  000000          OUT.13: .WORD 0                 ;DATA ARGUMENT 13
001517  000000          OUT.14: .WORD 0                 ;DATA ARGUMENT 14
001520  000000          OUT.15: .WORD 0                 ;DATA ARGUMENT 15
001521  000000          OUT.16: .WORD 0                 ;DATA ARGUMENT 16
001522  000000          OUT.17: .WORD 0                 ;DATA ARGUMENT 17
001523  000000          OUT.18: .WORD 0                 ;DATA ARGUMENT 18
001524  000000          OUT.19: .WORD 0                 ;DATA ARGUMENT 19
001525  000000          OUT.20: .WORD 0                 ;DATA ARGUMENT 20
001526  000000          OUT.21: .WORD 0                 ;DATA ARGUMENT 21
001527  000000          OUT.22: .WORD 0                 ;DATA ARGUMENT 22
001530  000000          OUT.23: .WORD 0                 ;DATA ARGUMENT 23
001531  000000          OUT.24: .WORD 0                 ;DATA ARGUMENT 24
001532  000000          OUT.25: .WORD 0                 ;DATA ARGUMENT 25
001533  000000          OUT.26: .WORD 0                 ;DATA ARGUMENT 26
001534  000000          OUT.27: .WORD 0                 ;DATA ARGUMENT 27
001535  000000          OUT.28: .WORD 0                 ;DATA ARGUMENT 28
001536  000000          OUT.29: .WORD 0                 ;DATA ARGUMENT 29
001537  000000          OUT.30: .WORD 0                 ;DATA ARGUMENT 30
001540  000000          OUT.31: .WORD 0                 ;DATA ARGUMENT 31
001541  000000          OUT.32: .WORD 0                 ;DATA ARGUMENT 32
001542  000000          OUT.33: .WORD 0                 ;DATA ARGUMENT 33
001543  000000          OUT.34: .WORD 0                 ;DATA ARGUMENT 34

                        ;IN BUFFER - DATA RECEIVED FROM HOST

001544  000000          IN.RQ:  .WORD 0                 ;HOST REQUEST CODE (ECHO)
001545  000000          IN.01:  .WORD 0                 ;DATA ARGUMENT  1
001546  000000          IN.02:  .WORD 0                 ;DATA ARGUMENT  2
001547  000000          IN.03:  .WORD 0                 ;DATA ARGUMENT  3
001550  000000          IN.04:  .WORD 0                 ;DATA ARGUMENT  4
001551  000000          IN.05:  .WORD 0                 ;DATA ARGUMENT  5
001552  000000          IN.06:  .WORD 0                 ;DATA ARGUMENT  6
001553  000000          IN.07:  .WORD 0                 ;DATA ARGUMENT  7
001554  000000          IN.08:  .WORD 0                 ;DATA ARGUMENT  8
001555  000000          IN.09:  .WORD 0                 ;DATA ARGUMENT  9
001556  000000          IN.10:  .WORD 0                 ;DATA ARGUMENT 10
001557  000000          IN.11:  .WORD 0                 ;DATA ARGUMENT 11
001560  000000          IN.12:  .WORD 0                 ;DATA ARGUMENT 12
001561  000000          IN.13:  .WORD 0                 ;DATA ARGUMENT 13
```

```
     58 001562  000000                IN.14:  .WORD 0                 ;DATA ARGUMENT 14
     59 001563  000000                IN.15:  .WORD 0                 ;DATA ARGUMENT 15
     60 001564  000000                IN.16:  .WORD 0                 ;DATA ARGUMENT 16
     61 001565  000000                IN.17:  .WORD 0                 ;DATA ARGUMENT 17
     62 001566  000000                IN.18:  .WORD 0                 ;DATA ARGUMENT 18
     63 001567  000000                IN.19:  .WORD 0                 ;DATA ARGUMENT 19
     64 001570  000000                IN.20:  .WORD 0                 ;DATA ARGUMENT 20
     65 001571  000000                IN.21:  .WORD 0                 ;DATA ARGUMENT 21
     66 001572  000000                IN.22:  .WORD 0                 ;DATA ARGUMENT 22
     67 001573  000000                IN.23:  .WORD 0                 ;DATA ARGUMENT 23
     68 001574  000000                IN.24:  .WORD 0                 ;DATA ARGUMENT 24
     69 001575  000000                IN.25:  .WORD 0                 ;DATA ARGUMENT 25
     70 001576  000000                IN.26:  .WORD 0                 ;DATA ARGUMENT 26
     71 001577  000000                IN.27:  .WORD 0                 ;DATA ARGUMENT 27
     72 001600  000000                IN.28:  .WORD 0                 ;DATA ARGUMENT 28
     73 001601  000000                IN.29:  .WORD 0                 ;DATA ARGUMENT 29
     74 001602  000000                IN.30:  .WORD 0                 ;DATA ARGUMENT 30
     75 001603  000000                IN.31:  .WORD 0                 ;DATA ARGUMENT 31
     76 001604  000000                IN.32:  .WORD 0                 ;DATA ARGUMENT 32
     77 001605  000000                IN.33:  .WORD 0                 ;DATA ARGUMENT 33
     78 001606  000000                IN.34:  .WORD 0                 ;DATA ARGUMENT 34
     79         000043                BUFSIZ  =       . - IN.RQ       ;SIZE OF BUFFER
```

```
                                        .SBTTL  STACK AREA
                                ;STACK AREA
                                
001607  123456                          .WORD 123456                    ;END MARKER FOR STACK
001610                                  .BLKW 31.                       ;STACK
001647  123456                  STACK:  .WORD 123456                    ;MARKER FOR STACK UNDERFLOW
```

```
                                        .SBTTL  GETU   - POLL ALL PORTS, THEN GET UNITS TO TEST
                                        ;GET WHICH UNITS TO TEST
001650                                  START:
                                        ;
                                        ;       GET THE UNITS TO TEST
                                        ;
                                        ;
                                        ;       POLL ALL PORTS AND FILL IN A UDA PORT INFORMATION TABLE (UNITS)
                                        ;
001650  104205  000001                          MOV     #1,R5           ; MOVE INITIAL MASK TO R5
001652  104204  000720                          MOV     #UNITS,R4       ; R4 POINTS TO UNIT TABLE
001654                                  5$:     PUSH    R4              ; SAVE R4
001654  100464                                                                  MOV R4,-(SP)
001655  104052                                  MOV     R5,R2           ; MOVE MASK TO R2
001656  024621                                  CALL    RDSTAT          ; GET DRIVE'S STATUS
001657  102201  010000                          BIT     #10000,R1       ; SEE IF ERROR
001661  011667                                  BEQ     10$             ; IF NOT, BRANCH
001662  104203  003772                          MOV     #SER10,R3       ; NO DRIVE ATTACHED
001664  100643  000001                          MOV     R3,1(R4)        ; SAVE ERROR MESSAGE
001666  002057                                  BR      85$             ; REPORT
001667  114003                          10$:    CLR     R3              ; SET UP TIMEOUT COUNT
001670  024621                          15$:    CALL    RDSTAT          ; GET STATUS
001671  102201  000001                          BIT     #RCVRDY,R1      ; SEE IF RECEIVER READY ASSERTED
001673  051703                                  BNE     20$             ; IF SO, BRANCH
001674  117403                                  DEC     R3              ; DECREMENT COUNT
001675  051670                                  BNE     15$             ; IF INCOMPLETE, BRANCH
001676  104203  004005                          MOV     #SER11,R3       ; RECEIVER READY NEVER ASSERTED
001700  100643  000001                          MOV     R3,1(R4)        ; SAVE ERROR MESSAGE
001702  002057                                  BR      85$             ; REPORT
001703  102201  000100                  20$:    BIT     #AVAIL,R1       ; SEE IF DRIVE IS AVAILABLE
001705  051713                                  BNE     25$             ; IF SO, BRANCH
001706  104203  004215                          MOV     #SER40,R3       ; GET SECONDARY ERROR
001710  100643  000001                          MOV     R3,1(R4)        ; SAVE
001712  002057                                  BR      85$             ; EXIT
001713  104202  001750                  25$:    MOV     #MAXSND,R2      ; SET UP MAXIMUM TRIES AT SENDING
001715  104203  001171                          MOV     #CR.GST,R3      ; R3 POINTS TO GET STATUS COMMAND
001717  104137                          30$:    MOV     (R3),R0         ; SET ADR OF SDI COMMAND BUFFER
001720  104631  000001                          MOV     1(R3),R1        ; SET BUFFER LENGTH
001722                                          PUSH    R2              ; SAVE R2
001722  100462                                                                  MOV R2,-(SP)
001723  104052                                  MOV     R5,R2           ; SETUP FOR SEND
001724  060004                                  XFC     SEND            ; SEND COMMAND
001725                                          POP     R2              ; RESTORE COUNT
001725  104262                                                                  MOV (SP)+,R2
001726  115001                                  TST     R1              ; DID UNIT ACCEPT COMMAND
001727  011737                                  BEQ     35$             ; IF SO, BRANCH
001730  117402                                  DEC     R2              ; DECREMENT COUNT
001731  051717                                  BNE     30$             ; IF UNEXPIRED, BRANCH
001732  104203  004023                          MOV     #SER12,R3       ; GET ERROR NUMBER
001734  100643  000001                          MOV     R3,1(R4)        ; SAVE
001736  002057                                  BR      85$
001737                                  35$:    PUSH    R4              ; SAVE R4
001737  100464                                                                  MOV R4,-(SP)
001740  104204  000003                          MOV     #3,R4           ; SET UP SHORT TIMEOUT
```

```
    54 001742  104637  000002          40$:    MOV     2(R3),R0        ; SET DATA BUFFER ADDRESS
    55 001744  104631  000003                  MOV     3(R3),R1        ; SET BUFFER LENGTH
    56 001746  104052                          MOV     R5,R2           ; SETUP FOR RECEIVE
    57 001747  060005                          XFC     RCV             ; RECEIVE SDI COMMAND
    58 001750  115001                          TST     R1              ; DID ERROR OCCUR
    59 001751  012011                          BEQ     70$             ; IF NOT, BRANCH
    60 001752  106201  000001                  CMP     #1,R1           ; SEE IF TIMEOUT
    61 001754  051763                          BNE     45$             ; IF NOT, BRANCH
    62 001755  117404                          DEC     R4              ; DECREMENT TIMEOUT VALUE
    63 001756  051742                          BNE     40$             ; IF NOT TIMEOUT, BRANCH
    64 001757                                  POP     R4              ; RESTORE R4
       001757  104264                                                                  MOV (SP)+,R4
    65 001760  104203  004035                  MOV     #SER13,R3       ; GET ERROR NUMBER
    66 001762  002006                          BR      65$             ; BRANCH TO EXIT
    67 001763                          45$:    POP     R4              ; RESTORE R4
       001763  104264                                                                  MOV (SP)+,R4
    68 001764  110601                          ROR     R1              ; ROTATE INTO POSITION TO TEST
    69 001765  110601                          ROR     R1              ; SEE IF FIRST WORD NOT START FRAME
    70 001766  041772                          BCC     50$             ; IF NOT, BRANCH
    71 001767  104203  004050                  MOV     #SER14,R3       ; GET ERROR NUMBER
    72 001771  002006                          BR      65$             ; BRANCH TO END OF LOOP
    73 001772  110601                  50$:    ROR     R1              ; SEE IF FRAMING ERROR
    74 001773  041777                          BCC     55$             ; IF NOT, BRANCH
    75 001774  104203  004076                  MOV     #SER15,R3       ; GET ERROR NUMBER
    76 001776  002006                          BR      65$             ; BRANCH TO END OF LOOP
    77 001777  110601                  55$:    ROR     R1              ; SEE IF CHECKSUM ERROR
    78 002000  042004                          BCC     60$             ; IF NOT, BRANCH
    79 002001  104203  004120                  MOV     #SER16,R3       ; GET ERROR NUMBER
    80 002003  002006                          BR      65$             ; BRANCH TO END OF LOOP
    81 002004  104203  004143          60$:    MOV     #SER17,R3       ; GET ERROR NUMBER
    82 002006  100643  000001          65$:    MOV     R3,1(R4)        ; SAVE
    83 002010  002057                          BR      85$             ; BRANCH TO END OF LOOP
```

```
                                        ;
                                        ;       NOW FILL IN THE TABLE WITH ALL THE SUBUNIT NUMBERS
                                        ;
002011                          70$:    POP     R4              ; RESTORE R4
002011  104264                                                                  MOV (SP)+,R4
002012  104207  177777                  MOV     #-1,R0          ; GET 'NO UNITS' FLAG
002014  100647  000001                  MOV     R0,1(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
002016  100647  000002                  MOV     R0,2(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
002020  104307  000744                  MOV     ST,R0           ; R0 HAS UNIT NUMBER
002022  104072                          MOV     R0,R2           ; COPY R0 TO R2
002023  103207  170000                  BIC     #^CHBHINB,R0    ; R0 HAS UNIT NUMBER
002025                                  PUSH    <R1,R2>         ; SAVE R1 AND R2
002025  100461                                                                  MOV R1,-(SP)
002026  100462                                                                  MOV R2,-(SP)
002027  104052                          MOV     R5,R2           ; MOVE UDA PORT MASK TO R2
002030  024621                          CALL    RDSTAT          ; GET STATUS
002031  102201  000002                  BIT     #ATTN,R1        ; SEE IF SPINABLE
002033  052036                          BNE     75$             ; IF SO, BRANCH
002034  101207  010000                  BIS     #10000,R0       ; SET 'NOT SPINABLE' FLAG
002036                          75$:    POP     <R2,R1>         ; RESTORE
002036  104262                                                                  MOV (SP)+,R2
002037  104261                                                                  MOV (SP)+,R1
002040  101207  040000                  BIS     #40000,R0       ; FLAG UNIT AS NOT TESTED
002042  110702                          SWAB    R2              ; SWAP R2'S BYTES
002043  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
002044  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
002045  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
002046  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
002047  103202  177760                  BIC     #LBLONB,R2      ; CLEAR ALL BUT SUBUNIT BITS
002051  100147                          MOV     R0,(R4)         ; MOVE UNIT NUMBER INTO UNITS
002052  110602                  80$:    ROR     R2              ; MOVE SUBUNIT BIT TO CARRY
002053  042057                          BCC     85$             ; IF NO MORE SUBUNITS, BRANCH
002054  100247                          MOV     R0,(R4)+        ; MOVE SUBUNIT NUMBER TO TABLE
002055  115407                          INC     R0              ; INCREMENT SUBUNIT NUMBER
002056  002052                          BR      80$             ; BRANCH
31
32
33
002057                          85$:    POP     R4              ; R4 POINTS TO START OF UNIT JUST HANDLED
002057  104264                                                                  MOV (SP)+,R4
002060  105204  000004                  ADD     #4,R4           ; R4 WILL POINT TO NEXT UNIT (CLEAR CARRY FOR ROL)
002062  110205                          ROL     R5              ; R5 HAS NEXT UNIT PORT MASK
002063  106205  000020                  CMP     #20,R5          ; SEE IF ALL PORTS TESTED
002065  051654                          BNE     5$              ; IF NOT, BRANCH
```

```
                                        ;
                                        ;       NOW GET THE PLUG NUMBERS TO TEST AND FIND THEM IN THE TABLE
                                        ;
002066  104207  060012                  MOV     #UTOTST,R0      ; GET WHAT SUBUNIT NUMBERS TO TEST REQUEST
002070  024652                          CALL    HOSTRQ          ; GET THE PLUG NUMBERS
002071  104207  001545                  MOV     #IN.01,R0       ; R0 POINTS TO UNIT NUMBERS TO TEST
002073  104201  000720          90$:    MOV     #UNITS,R1       ; R1 POINTS TO UDA PORT INFORMATION
002075  104112                  95$:    MOV     (R1),R2         ; R2 HAS UNIT
002076  072121                          BMI     115$            ; IF NONE, BRANCH
002077                                  PUSH    R1              ; SAVE R1
002077  100461                                                                  MOV R1,-(SP)
002100  103202  170000          100$:   BIC     #170000,R2      ; CLEAR 'NOT TESTED', DO NOT LEAVE 'UNSPINABLE' SET
002102  106172                          CMP     (R0),R2         ; SEE IF IT IS A UNIT TO TEST
002103  052107                          BNE     105$            ; NO MATCH
002104  100112                          MOV     R2,(R1)         ; SAVE UNIT AS ONE TO TEST
002105                                  POP     R1              ; RESTORE STACK
002105  104261                                                                  MOV (SP)+,R1
002106  002226                          BR      160$            ; LOOK FOR THE NEXT ONE
002107  115401          105$:   INC     R1              ; POINT TO NEXT SUBUNIT
002110  104012                          MOV     R1,R2           ; COPY TO R2
002111  107202  000720                  SUB     #UNITS,R2       ; SUBTRACT STARTING ADDRESS
002113  102202  000003                  BIT     #3,R2           ; SEE IF STILL ON SAME UNIT
002115  012120                          BEQ     110$            ; IF NOT, BRANCH
002116  104112                          MOV     (R1),R2         ; SEE IF ANY MORE SUBUNITS
002117  032100                          BPL     100$            ; IF SO, BRANCH
002120                          110$:   POP     R1              ; RESTORE R1
002120  104261                                                                  MOV (SP)+,R1
002121  105201  000004          115$:   ADD     #4,R1           ; LOOK AT NEXT UNIT
002123  106201  000740                  CMP     #UNITS+16.,R1   ; SEE IF ENTIRE TABLE SEARCHED
002125  052075                          BNE     95$             ; IF NOT, BRANCH
```

```
                                ;
                                ;       DIDN'T FIND THE REQUESTED UNITS -- DUMP ALL KNOWLEDGE OF THE
                                ;       UDA PORTS AND DIE
                                ;
002126  104170  001504          MOV     (R0),OUT.03     ; SAVE UNIT NUMBER IN REQUEST BUFFER
002130  104204  001506          MOV     #OUT.05,R4      ; R4 POINTS TO OUTPUT BUFFER
002132  104205  000720          MOV     #UNITS,R5       ; R5 POINTS TO UNIT TABLE
002134  104157          120$:   MOV     (R5),R0         ; GET FIRST WORD OF UNIT
002135  032142                  BPL     125$            ; IF VALID UNIT WAS FOUND, BRANCH
002136  104657  000001          MOV     1(R5),R0        ; GET POINTER TO ERROR MESSAGE
002140  100247                  MOV     R0,(R4)+        ; SAVE IN OUTPUT BUFFER
002141  002175                  BR      155$            ; EXIT
002142          125$:   PUSH    R5                      ; SAVE POINTER TO UNIT TABLE
002142  100465                                                          MOV R5,-(SP)
002143  102207  010000          BIT     #10000,R0       ; SEE IF DRIVE UNSPINABLE
002145  012151                  BEQ     130$            ; IF SPINABLE, BRANCH
002146  104207  004237          MOV     #SER41,R0       ; REPORT DRIVE(S) UNSPINABLE
002150  002153                  BR      135$            ; BRANCH
002151  104207  004170  130$:   MOV     #SER18,R0       ; GET POINTER TO ERROR MESSAGE
002153  100247          135$:   MOV     R0,(R4)+        ; SAVE
002154  114007                  CLR     R0              ; CLEAR COUNT
002155  104251          140$:   MOV     (R5)+,R1        ; GET UNIT NUMBER
002156  072163                  BMI     145$            ; IF INVALID, BRANCH
002157  115407                  INC     R0              ; INCREMENT COUNT
002160  106207  000004          CMP     #4,R0           ; SEE IF MAX
002162  052155                  BNE     140$            ; IF NOT, LOOP
002163  104671  001156  145$:   MOV     SER18E-1(R0),R1 ; GET POINTER TO CORRECT ERROR MESSAGE
002165  100241                  MOV     R1,(R4)+        ; MOVE INTO OUTPUT BUFFER
002166                          POP     R5              ; RESTORE R5
002166  104265                                                          MOV (SP)+,R5
002167                          PUSH    R5              ; SAVE R5
002167  100465                                                          MOV R5,-(SP)
002170  104251          150$:   MOV     (R5)+,R1        ; GET SUBUNIT NUMBER
002171  100241                  MOV     R1,(R4)+        ; SAVE
002172  117407                  DEC     R0              ; DECREMENT COUNT
002173  052170                  BNE     150$            ; IF COUNT INCOMPLETE, BRANCH
002174                          POP     R5              ; RESTORE R5
002174  104265                                                          MOV (SP)+,R5
002175  105205  000004  155$:   ADD     #4,R5           ; POINT TO NEXT UNIT TABLE
002177  106205  000740          CMP     #UNITS+16.,R5   ; SEE IF ENTIRE TABLE SEARCHED
002201  052134                  BNE     .120$           ; IF NOT, BRANCH
002202                          DEVFTL  5000            ; REPORT FATAL ERROR (WILL SHOW UP AS A 5000)
002202  104200  003653  001505                                  MOV     #ER5000,OUT.04
002205  104202  051610                                          MOV     #5000!FTLDEV,R2
002207  104020  001503                                          MOV     R2,OUT.02
002211  104200  002211  001502                                  MOV     #.,OUT.01
002214  104200  060014  001501                                  MOV     #ERRMC,OUT.RQ
002217  104307  001501          MOV     OUT.RQ,R0       ; SET UP FOR REPORT
002221  024652                  CALL    HOSTRQ          ; REPORT ERROR
002222  104207  060016          MOV     #DONE,R0        ; END TEST
002224  024652                  CALL    HOSTRQ          ; SEND END-OF-TEST TO HOST
002225  060021                  XFC     EXIT            ; TERMINATE DM

002226  115407  160$:   INC     R0              ; POINT TO NEXT UNIT TO TEST
002227  104172          MOV     (R0),R2         ; CHECK NEXT UNIT
002230  032073          BPL     90$             ; FIND IN UDA PORT INFORMATION
```

```
                                        .SBTTL  TEST 2 START TESTING
                                ;:+
                                ;       FIND OUT IF HOST HAD TO INIT UDA TO GET A PROGRAM FROM AN ATTACHED DRIVE.
                                ;       THE PROGRAM WAS REQUESTED BY THE DRIVE TO BE DOWNLINE LOADED INTO ITS MEMORY.
                                ;       THE INDICATION COMES FROM THE UTOTST RESPONSE.
                                ;
                                ;       A WORD WITH MSB IS SET TO INDICATE THE LAST UNIT ENTRY.  AFTER THAT WORD
                                ;       ANOTHER WORD INDICATES WHEATHER OR NOT A FILE WAS SUPPOSE TO BE DOWN LINE LOADED.
                                ;       A '0' MEANS NORMAL PROCESSING, NO FILE WAS EXPECTED OR REQUESTED.
                                ;       A '1' MEANS A FILE WAS EXPECTED AND IS AVAILABLE.
                                ;       A '2' MEANS A FILE WAS EXPECTED AND IS NOT AVAILABLE.
                                ;
                                ;       R0 IS A POINTER TO THE WORD BEYOND THE LAST TABLE ENTRY
                                ;       (R0) = 100000
                                ;:-
002231  104671  000001          T2STRT: MOV     1(R0),R1                ;WAS A FILE EXPECTED?
002233  012315                          BEQ     PORT0                   ;IF NOT, GO TO NORMAL PROCESSING
002234  115407                          INC     R0                      ;POINT TO INDICATOR IN IN.RQ BUFFER
002235  104075                          MOV     R0,R5                   ;R5 -> INDICATOR
002236  104151                          MOV     (R5),R1                 ;1ST WORD CONTAINS INDICATOR TO SEE IF A FILE WAS EX
002237  106201  000001                  CMP     #1,R1                   ;WAS THE EXPECTED FILE AVAILABLE?
002241  053534                          BNE     DO.DI7                  ;IF NOT AVAILABLE, GO TO REPORT ERROR
                                ; *** FILE WAS THERE, NOW SEE WHERE IN THE TABLE IT WAS SO PORT INDICATOR IS THE SAME?
002242  104207  000720                  MOV     #UNITS,R0               ;R0 -> UNIT TABLE
002244  104651  000001                  MOV     1(R5),R1                ;R1 = UNIT NUMBER
002246  104202  000001                  MOV     #UNIT0,R2               ;R2 = 1ST PORT
002250  114003                          CLR     R3                      ;R3 = UNIT TABLE INDICATOR
002251  114004                  1$:     CLR     R4                      ;R4 KEEPS TRACK OF WHICH SUBUNITS
002252  104270  001153          2$:     MOV     (R0)+,SAVREG            ;STORE UNIT IN SAVE REG
002254  072301                          BMI     4$                      ;IF NEGATIVE VALUE, DON'T COMPARE
002255  103300  177400  001153          BIC     HIBYTE,SAVREG           ;CLEAR EXTANEOUS BITS
002260  106301  001153                  CMP     SAVREG,R1               ;DID WE FIND THE DRIVE?
002262  012305                          BEQ     5$                      ;IF IT IS, WE HAVE FOUND IT
002263  115404                          INC     R4                      ;ELSE, INCREMENT SUBUNIT POINTER
002264  106204  000004                  CMP     #4,R4                   ;ENDED WITH THIS UNIT ENTRY?
002266  052252                          BNE     2$                      ;IF NOT, CONTINUE
002267  105022                  3$:     ADD     R2,R2                   ;ELSE, NEXT PORT SET IN R2
002270  115403                          INC     R3                      ;INCREMENT UNIT TABLE INDICATOR
002271  106203  000004                  CMP     #4,R3                   ;DONE?
002273  052251                          BNE     1$                      ;IF THIS GETS TO 4, THEN ERROR(UNIT NOT FOUND)
002274                                  ERRHRD  MS2000,R1
                                                                                MOV R1,-(SP)
C02274  100461                                                                  CALL RERROR     ;ERROR # 2000.
002275  025022                                                                  .WORD ERHARD+ERRN
002276  103720                                                                  .WORD <PRMS*10000>+MS2000
002277  010000
002300  002315                          BR      PORT0                   ;START REGULAR TESTING
                                ; *** IF HERE, NEGATIVE ENTRY, GO TO NEXT UNIT ENTRY.  ADJUST R0 TO PROPER POINTER
002301  105207  000003          4$:     ADD     #3,R0                   ;R0 -> INTO NEXT UNIT\INCREMENTED BY (R0)+
002303  107047                          SUB     R4,R0                   ;REALIGN TO POINT TO BEGINNING OF POINTER
002304  002267                          BR      3$                      ;GO BACK INTO THE LOOP
                                ; *** IF HERE, FOUND THE ENTRY, SAVE PERTINENT VALUES, GO DOWN LINE LOAD PROGRAM
002305  104650  000001  001152  5$:     MOV     1(R5),LUNIT             ;SAVE LOGICAL UNIT NUMBER
002310  104020  000741                  MOV     R2,SDI                  ;SAVE PORT VALUE
002312  104030  000740                  MOV     R3,UNITNB               ;SAVE UNIT NUMBER OFFSET IN TABLE
002314  003377                          BR      DO.DI2                  ;GO DOWN LINE LOAD PROGRAM
```

```
                                        .SBTTL  SEQUENCE THE DIAGNOSTICS TO ALL UNITS SELECTED
                                ;SEQUENCE THE DIAGNOSTICS TO ALL UNITS SELECTED.
                                ;TEST CODE WILL BE CALLED FOR EACH DISK SUBUNIT TO BE TESTED.
                                ;  LUNIT WILL CONTAIN LOGICAL UNIT NUMBER OF DRIVE FOR ERROR REPORTS
                                ;  SDI WILL CONTAIN SDI INTERCONNECT CODE FOR SELECTED DRIVE
                                ;  UNITNB WILL CONTAIN AN EVEN NUMBER FOR TESTING FIRST SUBUNIT OF A DRIVE
                                ;                      AN ODD NUMBER FOR TESTING SECOND SUBUNIT OF A DRIVE
                                
                                ; *** NORMAL PROCESSING
002315                          PORT0:
002315  114001                          CLR     R1              ;START WITH UNIT 0 INDEX
                                
002316  104207  000720          PORT2:  MOV     #UNITS,R0       ; GET POINTER TO UNITS TABLE
002320  105017                          ADD     R1,R0           ; ADD INDEX
002321  104173                          MOV     (R0),R3         ; GET CONTENTS OF TABLE
002322  032331                          BPL     1$              ; IF THIS UNIT IS PRESENT, BRANCH
002323  102201  000003                  BIT     #3,R1           ; SEE IF ON SUBUNIT 0 OF UNIT
002325  052363                          BNE     PORT5           ; IF NOT, TEST NEXT SUBUNIT
002326  105201  000003                  ADD     #3,R1           ; IF NO SUBUNIT 0, THEN NO UNIT - SKIP OTHER SUBUNITS
002330  002363                          BR      PORT5           ; BYPASS IF NO UNIT
002331  102203  040000          1$:     BIT     #40000,R3       ; SEE IF THIS UNIT IS TO BE TESTED
002333  052363                          BNE     PORT5           ; IF NOT, BRANCH
002334  104010  000740                  MOV     R1,UNITNB       ; STORE UNIT INDEX
002336  104030  001152                  MOV     R3,LUNIT        ; STORE LOGICAL UNIT NUMBER FOR DRIVE
002340  104202  000001                  MOV     #UNIT0,R2       ; GET UNIT 0 INTERCONNECT CODE
002342  110601                          ROR     R1              ; DIVIDE UNITNB BY FOUR
002343  110601                          ROR     R1
002344  103201  177760                  BIC     #LBLONB,R1      ; CLEAR UNUSED BITS
002346  117401          PORT3:  DEC     R1
002347  072354                          BMI     PORT4           ; FOR EACH DRIVE OVER 0 SHIFT R2 LEFT
002350  110202                          ROL     R2
002351  103202  000001                  BIC     #1,R2           ; CLEAR CARRY ROTATED INTO REG (IF ANY)
002353  002346                          BR      PORT3
002354  104020  000741          PORT4:  MOV     R2,SDI          ; STORE SDI INTERCONNECT CODE
002356  002373                          BR      TEST            ; PERFORM TEST ON THIS DRIVE
                                                                ; TEST RETURNS TO TESTX
002357  104206  001647          TESTX:  MOV     #STACK,SP       ; RESET STACK DO TO JUMPS OUT OF
                                                                ; SUBROUTINES
002361  104301  000740                  MOV     UNITNB,R1       ; GET UNIT INDEX
002363  115401          PORT5:  INC     R1              ; INCREMENT INDEX
002364  106201  000020                  CMP     #16.,R1         ; CHECK IF 16 DRIVES ALREADY SELECTED
002366  052316                          BNE     PORT2           ; REPEAT FOR ALL DRIVES
002367  104207  060016          DONECD: MOV     #DONE,R0        ; END OF PROGRAM
002371  024652                          CALL    HOSTRQ
002372  002367                          BR      DONECD          ; REPEAT IF RETURNED
```

```
                                .SBTTL  INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

                                ;START OF TEST CODE
                                ;
                                ;INPUTS:
                                ;       LUNIT - LOGICAL UNIT NUMBER OF DRIVE UNDER TEST
                                ;       SDI - SDI INTERCONNECT CODE FOR DRIVE

                                ;INITIALIZE THE DRIVE

002373  114003                 TEST:   CLR     R3                      ;R3 IS A DECREMENT COUNTER TO CHECK IF THE
                                                                        ; COMMAND HAS BEEN RECEIVED BY THE DRIVE
002374  104302  000741                 MOV     SDI,R2                  ;GET SDI SELECT CODE
002376  060011                         XFC     DINIT                   ;INITIALIZE THE DRIVE

                                ;WAIT FOR DRIVE TO ASSERT RECEIVER READY
                                ;TIME OUT AFTER ...?

002377  114005                         CLR     R5                      ;GET TIMEOUT COUNTER
002400  024621                 ILOOP:  CALL    RDSTAT                  ;GET DRIVE STATUS
002401  102201  010000                 BIT     #10000,R1               ;SEE IF ANY ERRORS
002403  012420                         BEQ     2$                      ;IF NO ERROR, BRANCH
002404  102201  004000                 BIT     #4000,R1                ;SEE IF XMIT ERROR
002406  117403                         DEC     R3                      ;ON SEND ERROR, DID WE DEPLETE THE COUNTER?
002407  052400                         BNE     ILOOP                   ;IF NOT, TRY AGAIN
002410                                 ERRHRD  MS2001                  ;REPORT INVALID STATUS ERROR
002410  025022                                                                 CALL RERROR     ;ERROR # 2001.
002411  103721                                                                 .WORD ERHARD+ERRN
002412  000043                                                                 .WORD <PRMS*10000>+MS2001
002413  002367                         BR      DONECD                  ;BRANCH TO DONE
002414                         1$:     ERRHRD  MS2002                  ;REPORT XMIT ERROR
002414  025022                                                                 CALL RERROR     ;ERROR # 2002.
002415  103722                                                                 .WORD ERHARD+ERRN
002416  000126                                                                 .WORD <PRMS*10000>+MS2002
002417  002367                         BR      DONECD                  ;BRANCH TO DONE
002420                         2$:
002420  114003                         CLR     R3                      ;R3 IS A DECREMENT COUNTER TO CHECK IF THE
                                                                        ; COMMAND HAS BEEN RECEIVED BY THE DRIVE
002421  102201  000001                 BIT     #RCVRDY,R1              ;CHECK RECEIVER READY LINE
002423  052431                         BNE     ECHO1                   ;ADVANCE IF ASSERTED
002424  117405                         DEC     R5                      ;DECREMENT TIME OUT COUNTER
002425  052400                         BNE     ILOOP                   ;STAY IN LOOP UNTIL SIGNAL SETS OR TIMEOUT
002426                                 ERRHRD  MS2003                  ;REPORT ERROR
002426  025022                                                                 CALL RERROR     ;ERROR # 2003.
002427  103723                                                                 .WORD ERHARD+ERRN
002430  000165                                                                 .WORD <PRMS*10000>+MS2003
```

```
                                .SBTTL  ECHO DATA TO DRIVE

                                ;ECHO THE FOLLOWING DATA PATTERNS TO THE DRIVE THEN CHECK THE
                                ;RESPONSE FOR THE PROPER DATA RECEIVED:
                                ;       377
                                ;       000
                                ;       252
                                ;       360
                                ;       017

002431  104205  001163      ECHO1:  MOV     #ECHOD,R5               ;GET POINTER TO ECHO DATA
002433  104157              ECHO1A: MOV     (R5),R0                 ;GET ECHO DATA
002434  103207  177400      ECHO2:  BIC     #HIBYTE,R0              ;CLEAR COMMAND FROM WORD
002436  060010                      XFC     ECHO                    ;PERFORM THE ECHO COMMAND

                                ;CHECK FOR TIMEOUT ERROR

002437  115001                      TST     R1                      ;CHECK FOR ERROR
002440  012460                      BEQ     ECHO4                   ;BRANCH IF NONE
002441  102201  000001              BIT     #1,R1                   ;CHECK IF SEND ERROR
002443  012453                      BEQ     ECHO3                   ;BRANCH IF RECEIVE ERROR
002444  117403                      DEC     R3                      ;IF SEND ERROR, IS THE COUNTER BEEN DEPLETED
002445  052433                      BNE     ECHO1A                  ;IF NOT, TRY AGAIN
002446                              ERRHRD  MS2004,R0               ;REPORT SEND ERROR
002446  100467                                                              MOV R0,-(SP)
002447  025022                                                              CALL RERROR     ;ERROR # 2004.
002450  103724                                                              .WORD ERHARD+ERRN
002451  010235                                                              .WORD <PRMS*10000>+MS2004
002452  002474                      BR      ECHO5
002453                  ECHO3:  ERRHRD  MS2005,R0               ;REPORT RECEIVE ERROR
002453  100467                                                              MOV R0,-(SP)
002454  025022                                                              CALL RERROR     ;ERROR # 2005.
002455  103725                                                              .WORD ERHARD+ERRN
002456  010275                                                              .WORD <PRMS*10000>+MS2005
002457  002474                      BR      ECHO5

                                ;CHECK DATA RECEIVED

002460  106157              ECHO4:  CMP     (R5),R0                 ;COMPARE DATA RECEIVED WITH
002461  012474                      BEQ     ECHO5                   ; DATA SENT
002462                              ERRHRD  MS2006,(R5),R0          ;REPORT DATA COMPARE ERROR
002462  100467                                                              MOV R0,-(SP)
002463  104010  001153                                                      MOV R1,SAVREG
002465  104151                                                              MOV (R5),R1
002466  100461                                                              MOV R1,-(SP,
002467  104301  001153                                                      MOV SAVREG,R1
002471  025022                                                              CALL RERROR     ;ERROR # 2006.
002472  103726                                                              .WORD ERHARD+ERRN
002473  020340                                                              .WORD <PRMS*10000>+MS2006

                                ;MOVE TO NEXT DATA PATTERN

002474  114003              ECHO5:  CLR     R3
002475  115405                      INC     R5                      ;BUMP TO NEXT DATA TO SEND
002476  104157                      MOV     (R5),R0                 ;CHECK IF AT END OF TABLE
002477  072434                      BMI     ECHO2                   ;SEND THIS DATA IF NOT
```

GET STATUS COMMAND

```
                                        .SBTTL  GET STATUS COMMAND

                                        ;ISSUE GET STATUS COMMAND AND CHECK THAT IT PERFORMS PROPERLY

002500  104200  000002  000742                MOV     #2,SAVSTA               ;COUNTER
002503  024140                                CALL    GTSTAT                  ;GET STATUS

                                        ;CLEAR DRIVE ERRORS

002504  024220                        DRCLR1: CALL    CLRDRV                  ; CLEAR DRIVE

                                        ;ISSUE GET STATUS COMMAND AND CHECK THAT IT PERFORMS PROPERLY

002505                                  GSTS3:
002505  024140                                CALL    GTSTAT                  ;GET STATUS

                                        ;LOOK AT DATA IN RESPONSE PACKET

002506  104307  000745                GSTS6:  MOV     ST+1,R0                 ;GET TWO WORDS FROM PACKET
002510  104301  000746                        MOV     ST+2,R1
002512  102201  000250                        BIT     #ST.WE+ST.PE+ST.FE,R1   ;CHECK ERROR BITS
002514  012524                              BEQ     10$                     ;BRANCH IF ALL CLEAR
002515  117400  000742                      DEC     SAVSTA                  ;TRY ONCE MORE?
002517  052504                            BNE     DRCLR1                  ;IF OK, TRY AGAIN
002520  024274                            CALL    STOSTA                  ;GO STORE STATUS
002521                                    ERRHRD  MS2007
002521  025022                                                                  CALL RERROR     ;ERROR # 2007.
002522  103727                                                                  .WORD ERHARD+ERRN
002523  000421                                                                  .WORD <PRMS*10000>+MS2007

002524  104070  000742                10$:    MOV     R0,SAVSTA               ;SAVE STATUS TO CHECK IF DIAGNOSTIC REQUEST BIT IS S

                                        ;ISSUE ONLINE COMMAND AND CHECK THAT IT PERFORMS PROPERLY

002526  104203  001446                        MOV     #CR.ONL,R3              ;R3 -> COMMAND
002530  024712                              CALL    TALKER                  ;SEND TO DRIVE
002531  115003                            TST     R3                      ;ALL OK?
002532  012547                            BEQ     30$                     ;IF SO, CHECK RESPONCE CODE
002533  032540                            BPL     20$                     ;ELSE, IS MSB SET?/IF NOT, RECEIVE ERROR
002534                                    ERRHRD  MS2008                  ;REPORT SEND ERROR
002534  025022                                                                  CALL RERROR     ;ERROR # 2008.
002535  103730                                                                  .WORD ERHARD+ERRN
002536  000500                                                                  .WORD <PRMS*10000>+MS2008
002537  002575                            BR      GSTS7                   ;
002540  024513                  20$:      CALL    TYPERR
002541                                    ERRHRD  MS2009,R0,R3            ;REPORT RECEIVE ERROR
002541  100463                                                                  MOV R3,-(SP)
002542  100467                                                                  MOV R0,-(SP)
002543  025022                                                                  CALL RERROR     ;ERROR # 2009.
002544  103731                                                                  .WORD ERHARD+ERRN
002545  020530                                                                  .WORD <PRMS*10000>+MS2009
002546  002575                            BR      GSTS7                   ;
002547  106207  000176          30$:      CMP     #COMPLT,R0              ;CHECK RESPONSE CODE
002551  012575                            BEQ     GSTS7                   ;IF OK, CONTINUE
002552  106207  000175                    CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
002554  052562                            BNE     35$                     ;IF NOT, UNRECOGNIZED RESPONSE
```

```
     47 002555  024274                          CALL    STOSTA                  ;CHECK STATUS
     48 002556                                  ERRHRD  MS2010                  ;
        002556  025022                                                                  CALL RERROR    ;ERROR # 2010.
        002557  103732                                                                  .WORD ERHARD+ERRN
        002560  000564                                                                  .WORD <PRMS*10000>+MS2010
     49 002561  002575                          BR      GSTS7
     50 002562                          35$:    ERRHRD  MS2011,#COMPLT,R0       ;IF NOT, REPORT ERROR
        002562  100467                                                                  MOV R0,-(SP)
        002563  104010  001153                                                          MOV R1,SAVREG
        002565  104201  000176                                                          MOV #COMPLT,R1
        002567  100461                                                                  MOV R1,-(SP)
        002570  104301  001153                                                          MOV SAVREG,R1
        002572  025022                                                                  CALL RERROR    ;ERROR # 2011.
        002573  103733                                                                  .WORD ERHARD+ERRN
        002574  020607                                                                  .WORD <PRMS*10000>+MS2011
     51
     52 002575                          GSTS7:
```

GET DRIVE CHARACTERISTICS

```
                                        .SBTTL  GET DRIVE CHARACTERISTICS
                                ;GET DRIVE CHARACTERISTICS
                                
002575  104200  000012  001154          MOV     #10.,SDISTO             ; SET UP TEMPORARY SHORT TIMEOUT VALUE
002600  104203  001440                  MOV     #CR.GCR,R3              ;POINT TO GET CHARS COMMAND
002602  024712                          CALL    TALKER                  ; INITIATE SDI INTERCHANGE
002603  115003                          TST     R3                      ;SEE IF ERROR OCCURRED
002604  012621                          BEQ     12$                     ;IF NOT, BRANCH
002605  032612                          BPL     11$                     ;IF SO, BRANCH
002606                                  ERRHRD  MS2012                  ;SEND COMMAND ERROR
002606  025022                                                                  CALL RERROR     ;ERROR # 2012.
002607  103734                                                                  .WORD ERHARD+ERRN
002610  000674                                                                  .WORD <PRMS*10000>+MS2012
002611  002647                          BR      T00
002612  024513                  11$:    CALL    TYPERR                  ;CHECK WHAT TYPE OF RECEIVE  RROR
002613                                  ERRHRD  MS2013,R0,R3
002613  100463                                                                  MOV R3,-(SP)
002614  100467                                                                  MOV R0,-(SP)
002615  025022                                                                  CALL RERROR     ;ERROR # 2013.
002616  103735                                                                  .WORD ERHARD+ERRN
002617  020735                                                                  .WORD <PRMS*10000>+MS2013
002620  002647                          BR      T00
002621  106207  000170          12$:    CMP     #CHRRES,R0              ;CHECK FOR SUCCESSFUL RESPONSE
002623  012647                          BEQ     T00
002624  106207  000175                  CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
002626  052634                          BNE     13$                     ;IF NOT, UNRECOGNIZED RESPONSE
002627  024274                          CALL    STOSTA                  ;CHECK STATUS
002630                                  ERRHRD  MS2014                  ;
002630  025022                                                                  CALL RERROR     ;ERROR # 2014.
002631  103736                                                                  .WORD ERHARD+ERRN
002632  001001                                                                  .WORD <PRMS*10000>+MS2014
002633  002647                          BR      T00
002634                          13$:    ERRHRD  MS2015,#CHRRES,R0       ;GET CHARACTERISTICS COMMAND FAILED
002634  100467                                                                  MOV R0,-(SP)
002635  104010  001153                                                          MOV R1,SAVREG
002637  104201  000170                                                          MOV #CHRRES,R1
002641  100461                                                                  MOV R1,-(SP)
002642  104301  001153                                                          MOV SAVREG,R1
002644  025022                                                                  CALL RERROR     ;ERROR # 2015.
002645  103737                                                                  .WORD ERHARD+ERRN
002646  021035                                                                  .WORD <PRMS*10000>+MS2015

002647  104307  000744          T00:    MOV     ST+SHRTTO,R0            ; GET SHORT TIMEOUT
002651  103207  177760                  BIC     #LBLONB,R0              ; CLEAR UNUSED BITS
002653  025005                          CALL    TO                      ; SET UP TIMEOUT
002654  104070  001154                  MOV     R0,SDISTO               ; SAVE IN SHORT TIMEOUT
002656  104307  000745                  MOV     ST+LONGTO,R0            ; GET LONG TIMEOUT
002660  103207  177760                  BIC     #LBLONB,R0              ; CLEAR UNUSED BITS
002662  025005                          CALL    TO                      ; SET UP TIMEOUT
002663  104070  001155                  MOV     R0,SDILTO               ; SAVE IN LONG TIMEOUT
```

```
                                                .SBTTL  CHECK WHICH COMMAND HAS BEEN GIVEN
002665  104307  000742          ST.DIA: MOV     SAVSTA,R0               ;GET STORED STATUS WORD
002667  102207  000040                  BIT     #ST.DR,R0               ;WAS THE DR BIT SET?
002671  053315                          BNE     DO.DIX                  ;IF SO, GO HANDLE IT WITHIN DIAGNOSE CODE
                                        ;; *** DO AN INITIAL WRITE MEMORY TO CLEAR THE REGION (ONLY 4 BYTES)
002672  104200  177774  001214          MOV     #FFFC,WRM+1             ;REGION ID
002675  114000  001215                  CLR     WRM+2                   ;OFFSET
002677  104200  000004  001216          MOV     #4,WRM+3                ;BYTE COUNT AND 1ST DATA BYTE
002702  114000  001217                  CLR     WRM+4                   ;DATA
002704  114000  001220                  CLR     WRM+5                   ;DATA
002706  104200  000012  001207          MOV     #10.,CR.WRM+1           ;SET COMMAND BYTE COUNT
002711  023730                          CALL    WRTMEM                  ;CLEAR BUFFER
002712  104200  000007  001207          MOV     #7,CR.WRM+1             ;SET COMMAND BYTE COUNT
                                        ; *** NOW GO DIAGNOSE REGION ZERO
002715  114000  001502                  CLR     OUT.01                  ;CLEAR OUT BUFFER TO INITIAL DIAGNOSE COMMANDS
002717  114000  001503                  CLR     OUT.02                  ;
002721  114000  001144                  CLR     DIAG.1                  ;MAKE SURE DIAGNOSE COMMAND RETURNS OP-CODE = 0
                                                                        ; (FOR 1ST TIME THROUGH ONLY)
002723  114000  001546                  CLR     IN.02                   ;CLEAR REGION ID FOR 1ST DIAGNOSE COMMAND
002725  003055                          BR      DIAGNS                  ;DO DIAGNOSE COMMAND, 1ST THING
                                        ;
                                        ; *** RETURN HERE TO GET NEW COMMAND
002726                          GT.CMD:                                 ;
002726  104207  060002                  MOV     #T2CMD,R0               ;SET REQUEST NUMBER
002730  024652                          CALL    HOSTRQ                  ;ASK HOST FOR PORTS
002731  104307  001545                  MOV     IN.01,R0                ;R0 = RESPONSE CODE;IN.02
                                                                        ; 0 = EXIT ; 1 = WRITE
                                                                        ; 2 = READ ; 3 = DIAGNOSE
002733  013101                          BEQ     TESTEX                  ; OP = 0 /EXIT
002734  117407                          DEC     R0                      ; R0 = 1?
002735  012755                          BEQ     WRITE                   ; IF SO, WRITE
002736  117407                          DEC     R0                      ; R0 = 2?
002737  013021                          BEQ     READ                    ; IF SO, READ
002740  117407                          DEC     R0                      ; R0 = 3?
002741  013075                          BEQ     DIAGNO                  ; IF SO, DIAGNOSE
002742                                  ERRHRD  MS2016,IN.01            ; ELSE, ERROR=INVALID INPUT
    002742  104010  001153                                                      MOV R1,SAVREG
    002744  104301  001545                                                      MOV IN.01,R1
    002746  100461                                                              MOV R1,-(SP)
    002747  104301  001153                                                      MOV SAVREG,R1
    002751  025022                                                              CALL RERROR     ;ERROR # 2016.
    002752  103740                                                              .WORD ERHARD+ERRN
    002753  011133                                                              .WORD <PRMS*10000>+MS2016
002754  002726                          BR      GT.CMD
```

```
                                                .SBTTL  MEMORY WRITE
                                        ;:+
                                        ;
                                        ;       INPUT   IN.02 = REGION ID
                                        ;               IN.03 = OFFSET
                                        ;               IN.04 = DATA BYTE
                                        ;
                                        ;       OUTPUT  OUT.RQ HAS DAT SET FOR T2CMD
                                        ;:-
002755                                  WRITE:
002755  104300  001546  001214                MOV     IN.02,WRM+1             ;REGION ID
002760  104300  001547  001215                MOV     IN.03,WRM+2             ;OFFSET
002763  104307  001550                        MOV     IN.04,R0                ;R0 = BYTE OF DATA IN LO BYTE
002765  110707                                SWAB    R0                      ;R0 IS IN HI BYTE
002766  103207  000377                        BIC     #LOBYTE,R0              ;CLEAR LO BYTE
002770  101207  000001                        BIS     #1,R0                   ;SET BYTE COUNT
002772  104070  001216                        MOV     R0,WRM+3                ;SET WORD IN PACKET
002774  114000  001217                        CLR     WRM+4
002776  114000  001220                        CLR     WRM+5
003000  104200  000007  001207                MOV     #7,CR.WRM+1             ;SET COMMAND BYTE COUNT
                                        ;
003003  023730                                CALL    WRTMEM
003004  104200  000001  001503                MOV     #1,OUT.02               ;R0 = OP CODE
003007  104300  001152  001502                MOV     LUNIT,OUT.01            ;DRIVE NUMBER
003012  114000  001504                        CLR     OUT.03                  ;
003014  114000  001505                        CLR     OUT.04                  ;
003016  114000  001506                        CLR     OUT.05                  ;
003020  002726                                BR      GT.CMD                  ;GO SEND REQUEST
```

MEMORY READ

```
                                        .SBTTL  MEMORY READ
                                ;++
                                ;
                                ;       INPUT   IN.02 HAS REGION ID
                                ;               IN.03 HAS OFFSET
                                ;
                                ;       OFFSET  SETS UP OUTBUFFER FOR T2CMD
                                ;--
003021                          READ:
003021  104300  001546  001430          MOV     IN.02,RDM+1             ;REGION ID
003024  104300  001547  001431          MOV     IN.03,RDM+2             ;OFFSET
003027  104200  000001  001432          MOV     #1,RDM+3                ;BYTE COUNT
                                ;
003032  023642                          CALL    RDMEM
                                ; *** SET UP RESPONSE PACKET
003033  104300  001152  001502          MOV     LUNIT,OUT.01            ;SET UP REQUEST
003036  104200  000002  001503          MOV     #2,OUT.02               ;RETURN OPCODE SET IN BUFFER
003041  104307  000744                  MOV     ST,R0                   ;R0 = BYTE COUNT + DATA
003043  110707                          SWAB    R0                      ;DATA IN LO BYTE
003044  103207  177400                  BIC     #HIBYTE,R0              ;CLEAR BYTE COUNT (1 BYTE ONLY SENT)
003046  104070  001504                  MOV     R0,OUT.03               ;STORE IN BUFFER FOR HOST
003050  114000  001505                  CLR     OUT.04                  ;
003052  114000  001506                  CLR     OUT.05                  ;
003054  002726                          BR      GT.CMD                  ;GO SEND REQUEST
```

```
                                                .SBTTL  SEND DIAGNOSE COMMAND
003055  102200  000001  000742  DIAGNS: BIT     #ST.RU,SAVSTA           ;CHECK IF DRIVE IS SPINNABLE (RUN SWITCH IN?)
003060  013075                          BEQ     3$                      ;ONLY DO DIAGNOSE ONCE IF NOT SPINNABLE
                                ; *** DRIVE IS SPINNABLE
003061  104302  000741                  MOV     SDI,R2                  ;R2 HAS CURRENT SDI PORT INDEX
003063  102200  000020  000742          BIT     #ST.SR,SAVSTA           ;HAS THE DRIVE BEEN SPUN UP?
003066  053074                          BNE     2$                      ;IF NOT ALREADY SPUN UP, DO DIAGNOSE TWICE
                                ; *** DRIVE HAS NOT BEEN SPUN UP
003067  023102                  1$:     CALL    DIAGDR                  ;DO A DIAGNOSE COMMAND
003070  104302  000741                  MOV     SDI,R2                  ;RESTORE PORT INDEX
003072  024010                          CALL    RUNDR                   ;SPIN UP THE DRIVE
003073  003075                          BR      3$
                                ; *** DRIVE HAS BEEN SPUN UP
003074  024064                  2$:     CALL    RECAL                   ;ELSE DO A RECAL
003075                          3$:
003075  104302  000741          DIAGNO: MOV     SDI,R2                  ;RESTORE PORT INDEX
003077  023102                          CALL    DIAGDR                  ;DO A DIAGNOSE COMMAND
003100  002726                          BR      GT.CMD                  ;GET COMMAND

                                ;END OF TESTING THIS DRIVE

003101  002357                  TESTEX: BR TESTX                        ;GO BACK TO DRIVE SEQUENCER
```

```
                                        .SBTTL  TEST 2 SPECIFIC ROUTINES
                                        .SBTTL  DIAGNOSE COMMAND PROCESSING
                                ;;+
                                ;
                                ;       INPUT   IN.02 HAS REGION ID
                                ;
                                ;       OUTPUT  OUT.RQ IS SET UP FOR T2CMD
                                ;               IF ERROR, R2 = 1
                                ;                   ELSE, R2 = 0
                                ;;-
003102                          DIAGDR:
003102  104200  177777  001151        MOV     #177777,SENDHR          ;SET SENDHR NONZERO FOR DIAGNOSE COMMAND
003105  104300  001546  001463        MOV     IN.02,DIA+1             ;SET UP MEMORY REGION ID
003110  104203  001455                MOV     #CR.DIA,R3              ;R3->DIAGNOSE PACKET
003112  104302  000741                MOV     SDI,R2                  ;R2 = PORT
003114  024712                        CALL    TALKER                  ;SEND COMMAND AND RECEIVE RESPONSE FROM DRIVE
003115  114000  001151                CLR     SENDHR
003117  115003                        TST     R3                      ;SEE IF ERROR OCCURRED
003120  013136                        BEQ     2$                      ;IF NO ERRORS, BRANCH
003121  110203                        ROL     R3                      ;SHIFT HIGH BIT INTO CARRY BIT
003122  043127                        BCC     1$                      ;IF CARRY CLEAR (RECEIVE ERROR) BRANCH
                                ; *** REPORT TRANSMISSION ERROR
003123                                  ERRHRD  MS2017
003123  025022                                                          CALL RERROR     ;ERROR # 2017.
003124  103741                                                          .WORD ERHARD+ERRN
003125  001224                                                          .WORD <PRMS*10000>+MS2017
003126  003167                          BR      4$
                                ; *** REPORT RECEPTION ERROR
003127  024513                  1$:     CALL    TYPERR
003130                                  ERRHRD  MS2018,R0,R3
003130  100463                                                          MOV R3,-(SP)
003131  100467                                                          MOV R0,-(SP)
003132  025022                                                          CALL RERROR     ;ERROR # 2018.
003133  103742                                                          .WORD ERHARD+ERRN
003134  021255                                                          .WORD <PRMS*10000>+MS2018
003135  003167                          BR      4$
                                ; *** CHECK RESPONSE OP-CODE FROM DRIVE
003136                          2$:
003136  106207  000374                CMP     #DIA.EN,R0              ;CHECK RESPONSE CODE
003140  013173                        BEQ     5$                      ;IF EQUAL, BRANCH
003141  106207  000176                CMP     #COMPLT,R0              ; WAS IT COMPLETED?
003143  013173                        BEQ     5$                      ; IF SO, BRANCH
003144  106207  000175                CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
003146  053154                        BNE     3$                      ;IF NOT, UNRECOGNIZED RESPONSE
003147  024274                        CALL    STOSTA                  ;CHECK STATUS
003150                                  ERRHRD  MS2019                  ;
003150  025022                                                          CALL RERROR     ;ERROR # 2019.
003151  103743                                                          .WORD ERHARD+ERRN
003152  001312                                                          .WORD <PRMS*10000>+MS2019
003153  003167                          BR      4$
003154                          3$:     ERRHRD  MS2020,#DIA.EN,R0
003154  100467                                                          MOV R0,-(SP)
003155  104010  001153                                                  MOV R1,SAVREG
003157  104201  000374                                                  MOV #DIA.EN,R1
003161  100461                                                          MOV R1,-(SP)
003162  104301  001153                                                  MOV SAVREG,R1
003164  025022                                                          CALL RERROR     ;ERROR # 2020.
```

```
            003165  103744                                          .WORD ERHARD+ERRN
            003166  021336                                          .WORD <PRMS*10000>+MS2020
     41     003167  024220                      4$:     CALL    CLRDRV
     42     003170  104202  000001                    MOV     #1,R2
     43     003172  000000                            RETURN
```

```
                                        .SBTTL  DIAGNOSE/READ MEMORY TO SEE IF ERROR OCCURRED
                                ; *** DO A READ MEMORY SDI COMMAND
003173  114000  001151          5$:     CLR     SENDHR
003175  104300  000744  001430          MOV     ST,RDM+1                ;REGION ID SET
003200  114000  001431                  CLR     RDM+2                   ;CLEAR OFFSET
003202  104200  000010  001432          MOV     #8.,RDM+3               ;BYTE COUNT = 8
003205  023642                          CALL    RDMEM                   ;READ THE MEMORY
003206  115002                          TST     R2                      ;ALL OK?
003207  053265                          BNE     DO.DC4                  ;IF NOT, DO DRIVE CLEAR/TRY NEXT DRIVE
003210  104302  000741                  MOV     SDI,R2                  ;RESTORE PORT INDICATOR
003212  104204  001506                  MOV     #OUT.05,R4              ;R4 -> OUT BUFFER
003214  024325                          CALL    CONMEM                  ;CONVERT MEMORY
                                ; *** DATA NOW IN OUT.RQ
003215  104303  001507                  MOV     OUT.06,R3               ;R3 HAD ET & DA FLAGS
                                ; *** CHECK IF DATA IS AVAILABLE FOR INFORMATION
003217  102203  040000                  BIT     #DATAVL,R3              ;IS DATA AVAILABLE?
003221  013236                          BEQ     6$                      ;IF NOT, CONTINUE WITH THIS DRIVE
003222  104200  000010  001431          MOV     #8.,RDM+2               ;PREVIOUS BYTE COUNT
003225  024365                          CALL    GETBCN                  ;GET NEW BYTE COUNT
003226  023642                          CALL    RDMEM                   ;READ MEMORY XFC
003227  115002                          TST     R2                      ;ALL OK?
003230  053265                          BNE     DO.DC4                  ;IF NOT, DO DRIVE CLEAR/ELSE, DROP DRIVE
003231  104302  000741                  MOV     SDI,R2                  ;RESTORE PORT INDICATOR
003233  104204  001512                  MOV     #OUT.09,R4              ;R4 -> OUT BUFFER
003235  024325                          CALL    CONMEM                  ;CONVERT MEMORY
003236  102203  100000          6$:     BIT     #ERRTYP,R3              ;WAS ET SET? WITH NO ERROR NUMBER???
003240  013245                          BEQ     7$                      ;IF NOT, CONTINUE
                                ; *** REPORT AN ERROR
003241                                  ERRHRD  MS2021
003241  025022                                                          CALL RERROR     ;ERROR # 2021.
003242  103745                                                          .WORD ERHARD+ERRN
003243  001424                                                          .WORD <PRMS*10000>+MS2021
003244  003265                          BR      DO.DC4                  ;GO DO DRIVE CLEAR
003245  102203  040000          7$:     BIT     #DATAVL,R3              ;IS DATA AVAILABLE?
003247  013265                          BEQ     DO.DC4                  ;IF NOT, DO DRIVE CLEAR
003250                                  MSSGE   MSG1
003250  104300  001152  001502                          MOV     LUNIT,OUT.01
003253  104200  003242  001503                          MOV     #MSG1,OUT.02
003256  100467                                                  MOV R0,-(SP)
003257  100461                                                  MOV R1,-(SP)
003260  104207  060015                          MOV     #MESSAG,R0
003262  024652                                  CALL    HOSTRQ
003263  104261                                                  MOV (SP)+,R1
003264  104267                                                  MOV (SP)+,R0
```

```
                                        .SBTTL  DIAGNOSE/DO A DRIVE CLEAR
                                ; *** DO DRIVE CLEAR, GET STATUS AND CHECK IF DR BIT IS SET
003265  104200  000004  001156  DO.DC4: MOV     #4,SAFWRD               ;SAFWRD = # MAXIMUM DRIVE CLEAR TRIES
003270  104302  000741          DO.DCL: MOV     SDI,R2                  ;RESTORE PORT INDICATOR
003272  024220                          CALL    CLRDRV                  ;CALL DRIVE CLEAR
003273  115003                          TST     R3                      ;ERROR?
003274  013301                          BEQ     1$                      ;IF NOT, CONTINUE
003275  117400  001156                  DEC     SAFWRD                  ;REACHED MAXIMUM # OF DRV CLR TRIES?
003277  053270                          BNE     DO.DCL                  ;IF NOT, TRY AGAIN
003300  003612                          BR      DO.DIO                  ; ELSE, ERROR
003301  024140                  1$:     CALL    GTSTAT                  ;CALL GET STATUS
003302  115003                          TST     R3                      ;ERROR?
003303  013310                          BEQ     2$                      ;IF NOT, CONTINUE
003304  117400  001156                  DEC     SAFWRD                  ;REACHED MAXIMUM # OF DRV CLR TRIES?
003306  053270                          BNE     DO.DCL                  ;IF NOT, TRY AGAIN
003307  003612                          BR      DO.DIO                  ; ELSE, ERROR
003310  104307  000745          2$:     MOV     ST+1,R0                 ;GET STATUS WORD
003312  102207  000040                  BIT     #ST.DR,R0               ;DR SET?
003314  013612                          BEQ     DO.DIO                  ;IF NOT, EXIT
                                ; *** IF DR BIT IS SET, READ 6 BYTES FROM REGION ID FFFD
003315  104200  177775  001430  DO.DIX: MOV     #FFFD,RDM+1             ;FFFD (REGION ID) STORED IN READ MEM PACKET
003320  114000  001431                  CLR     RDM+2                   ;OFFSET = 0
003322  104200  000006  001432          MOV     #6,RDM+3                ;BYTE COUNT = 6
003325  023642                          CALL    RDMEM                   ;READ MEMORY
003326  115002                          TST     R2                      ;WAS THERE AN ERROR?
003327  013331                          BEQ     3$                      ;IF THERE WAS NOT, CONTINUE
003330  C03612                          BR      DO.DIO                  ;DO A DRIVE CLEAR
003331  104302  000741          3$:     MOV     SDI,R2                  ;RESTORE PORT INDICATOR
003333  104204  000744                  MOV     #ST,R4                  ;R4 -> ST
003335  024325                          CALL    CONMEM                  ;CONVERT MEMORY
                                ; *** GET REGION ID AND PROGRAM NAME IF ANY
003336  104300  000744  000743          MOV     ST,SAVRID               ;SAVE REGION ID
003341  115000  000745                  TST     ST+1                    ;IS CHARACTER ENCODED?
003343  053347                          BNE     DO.DI1                  ;IF SO, GO GET FILE FROM HOST
003344  115000  000746                  TST     ST+2                    ;IF 1ST WORD 0, IS THE SECOND?
003346  013530                          BEQ     DO.DI3                  ;IF SO, GO DIAGNOSE REGION
```

```
                                                .SBTTL  DIAGNOSE/GET PROGRAM NAME SPECIFIED BY DRIVE AND DOWNLINE LOAD
                                        ; *** GET THE PROGRAM WHICH NAME WAS SPECIFIED BY THE DRIVE
003347                                  DO.DI1:
003347  104300  000743  001214                  MOV     SAVRID,WRM+1            ;STORE REGION ID IN THE WRITE PACKET
003352  104203  001502                          MOV     #OUT.01,R3              ;R3 -> INPUT DATA
003354  104307  001152                          MOV     LUNIT,R0
003356  100237                                  MOV     R0,(R3)+                ;SET UNIT NUMBER
003357  114007                                  CLR     R0
003360  100237                                  MOV     R0,(R3)+                ;CLEAR VALUE
003361  104307  000744                          MOV     ST,R0
003363  100237                                  MOV     R0,(R3)+                ;SAVE REGION ID
003364  104307  000745                          MOV     ST+1,R0
003366  100237                                  MOV     R0,(R3)+                ;1ST HALF NAME SAVED
003367  104307  000746                          MOV     ST+2,R0
003371  100237                                  MOV     R0,(R3)+                ;2ND HALF NAME SAVED
003372  104207  060001                          MOV     #T2DLL,R0               ;SET R0 = TEST 2 DOWNLINE LOAD
003374  024652                                  CALL    HOSTRQ
                                        ; *** IF HERE, DIDN'T REINIT UDA TO GET DATA
003375  104205  001545                          MOV     #IN.01,R5               ;R5 -> INPUT BUFFER
                                        ; *** DOWN LINE LOAD PROGRAM
003377  114000  001215                  DO.DI2: CLR     WRM+2                   ;CLEAR OFFSET
003401  104157                                  MOV     (R5),R0                 ;IS THE PROGRAM THERE?
003402  117407                                  DEC     R0
003403  053537                                  BNE     DO.DI8                  ;IF NOT, REPORT ERROR
                                        ; *** GET THE PROGRAM FROM THE HOST
003404  104657  000004                  1$:     MOV     4(R5),R0                ;UNIBUS ADDRESS PA
003406  104651  000005                          MOV     5(R5),R1                ;UNIBUS ADDRESS EA
003410  104652  000006                          MOV     6(R5),R2                ;BYTE COUNT
003412  053416                                  BNE     2$                      ;IF BYTE COUNT IS NOT ZERO, CONTINUE
003413                                          ERRHRD  MS2022                  ;ELSE, REPORT THE ERROR
003413  025022                                                                          CALL RERROR     ;ERROR # 2022.
003414  103746                                                                          .WORD ERHARD+ERRN
003415  001511                                                                          .WORD <PRMS*10000>+MS2022
003416  104203  000744                  2$:     MOV     #ST,R3                  ;POINTER TO INPUT BUFFER
003420  106202  000200                          CMP     #STSIZE,R2              ;R2 > MAX BUFFER SIZE?
003422  033425                                  BPL     3$                      ;IF NOT, CONTINUE
003423  104202  000200                          MOV     #STSIZE,R2              ;SET MAX BUFFER SIZE IN R2
003425  060013                          3$:     XFC     UREAD                   ;DO A UNIBUS READ
                                        ; *** SET UP SDI COMMAND PACKETS
003426  104020  001207                          MOV     R2,CR.WRM+1             ;SET BYTE COUNT IN WRM PACKET
003430  105200  000006  001207                  ADD     #6,CR.WRM+1             ;ADD TO COUNT TO INCLUDE PACKET LENGTH(1),
                                                                                ; SDI OPCODE(1), REGION ID(2), AND OFFSET(2)
003433  104650  000003  001214                  MOV     3(R5),WRM+1             ;SET REGION ID
003436  104020  001216                          MOV     R2,WRM+3                ;SET BYTE COUNT IN BUFFER
003440  104303  000744                          MOV     ST,R3                   ;1ST DATA WORD IN R3
003442  110703                                  SWAB    R3                      ;1ST DATA WORD IN UPPER BYTE
003443  103203  000377                          BIC     #LOBYTE,R3              ;CLEAR LOW BYTE
003445  101030  001216                          BIS     R3,WRM+3                ;SET DATA IN 1ST BUFFER WORD OF WRT MEM PACKET
003447  104020  001153                          MOV     R2,SAVREG               ;SAVE BYTE COUNT
003451  117402                                  DEC     R2                      ;ADJUST BYTE COUNT
003452  103200  000377  000744                  BIC     #LOBYTE,ST              ;CLEAR LOW BYTE OF 1ST DATA WORD OF PROGRAM
003455  101020  000744                          BIS     R2,ST                   ;REPLACE IT BY THE BYTE COUNT OF THE REST
003457  104204  001217                          MOV     #WRM+4,R4               ;R4 -> OUT BUFFER
003461  024325                                  CALL    CONMEM                  ;CONVERT MEMORY
                                        ; *** SEND TO THE DRIVE
003462  104302  000741                          MOV     SDI,R2                  ;RESTORE PORT INDICATOR
003464  023730                                  CALL    WRTMEM                  ;SEND TO DRIVE
```

```
     55 003465  115002                          TST     R2                      ;ALL OK?
     56 003466  053265                          BNE     DO.DC4                  ;IF NOT, GO CLEAR DRIVE
     57 003467  104302  001153                  MOV     SAVREG,R2               ;R2 = BYTE COUNT
     58 003471  105020  001215                  ADD     R2,WRM+2                ;ADJUST OFFSET
     59 003473  104657  000006                  MOV     6(R5),R0
     60 003475  107027                          SUB     R2,R0
     61 003476  100657  000006                  MOV     R0,6(R5)                ;DECREMENT TOTAL BYTE COUNT/DONE?
     62 003500  013517                          BEQ     5$                      ;IF 0, EXIT
     63 003501  073517                          BMI     5$                      ;IF NEG, EXIT
     64 003502  104657  000004                  MOV     4(R5),R0
     65 003504  105027                          ADD     R2,R0
     66 003505  100657  000004                  MOV     R0,4(R5)                ;ELSE, ADJUST UNIBUS ADDRESS TO READ FROM
     67 003507  106027                          CMP     R2,R0                   ;IS PA ADDRESS LESS THAN BUFFER SIZE?
     68 003510  073404                          BMI     1$                      ;IF IT IS NOT, CONTINUE
     69 003511  104657  000005                  MOV     5(R5),R0
     70 003513  115407                          INC     R0                      ;ELSE, INCREMENT EA ADDRESS (CROSSED 0 BOUNDARY)
     71 003514  100657  000005                  MOV     R0,5(R5)
     72 003516  003404                  4$:     BR      1$                      ;READ IN NEXT BUFFER
     73
     74                                 ; *** SET UP DIAGNOSE COMMAND
     75 003517  114000  001215          5$:     CLR     WRM+2                   ;REINIT OFFSET
     76 003521  104200  000007  001207          MOV     #7,CR.WRM+1             ;REINIT BYTE COUNT OF INSTRUCTION
     77 003524  104650  000003  001546          MOV     3(R5),IN.02             ;SET UP REGION ID
     78 003527  003102                          BR      DIAGDR                  ;SEND DIAGNOSE COMMAND
     79
     80                                 ; *** JUST RECEIVED THE REGION ID TO DIAGNOSE TO.  GO DIAGNOSE IT.
     81 003530  104300  000743  001546  DO.DI3: MOV     SAVRID,IN.02            ;STORE REGION ID FOR DIAGNOSE COMMAND
     82 003533  003102                          BR      DIAGDR                  ;GO DIAGNOSE
```

```
      1                                                .SBTTL  DIAGNOSE/REPORT ERROR -- NO DOWNLINE LOAD PROGRAM
      2                                          ; *** NO DOWN LINE LOAD PROGRAM AFTER REINITED UDA
      3 003534  104650  000001  000740  DO.DI7: MOV     1(R5),UNITNB            ;SET UP LOGICAL UNIT NUMBER
      4
      5                                          ; *** NO DOWN LINE LOAD PROGRAM, REPORT ERROR
      6 003537  104654  000007          DO.DI8: MOV     7(R5),R4                ;R4 = 1ST WORD OF PROGRAM NAME
      7 003541  024553                          CALL    DIV50                   ;DIVIDE BY 50 AND GET 3RD CHARACTER
      8 003542  110707                          SWAB    R0                      ;PUT IN PROPER BYTE
      9 003543  104070  001146                 MOV     R0,NAM.2                ;SET 3RD CHARACTER
     10 003545  024553                          CALL    DIV50                   ;GET 2ND CHARACTER
     11 003546  104070  001147                 MOV     R0,NAM.3                ;SET 2ND CHARACTER
     12 003550  104047                          MOV     R4,R0                   ;R0 = RAD50 OF 1ST CHARACTER
     13 003551  024567                          CALL    FNDASC                  ;GET ASCII EQUIVALENT
     14 003552  110707                          SWAB    R0                      ;PUT IN PROPER BYTE
     15 003553  101070  001147                 BIS     R0,NAM.3                ;SET 1ST CHARACTER
     16 003555  104654  000010                 MOV     10(R5),R4               ;R4 = 2ND WORD OF PROGRAM NAME
     17 003557  024553                          CALL    DIV50                   ;GET 6TH CHARACTER
     18 003560  104070  001145                 MOV     R0,NAM.1                ;SET 6TH CHARACTER
     19 003562  024553                          CALL    DIV50                   ;GET 5TH CHAPACTER
     20 003563  110707                          SWAB    R0                      ;PUT IN PROPER BYTE
     21 003564  101070  001145                 BIS     R0,NAM.1                ;SET 5TH CHARACTER
     22 003566  104047                          MOV     R4,R0                   ;R0 = RAD50 OF 4TH CHARACTER
     23 003567  024567                          CALL    FNDASC                  ;FIND ASCII EQUIVALENT
     24 003570  101070  001146                 BIS     R0,NAM.2                ;SET 4TH CHARACTER
     25 003572                                  ERRHRD  MS2023,NAM.1,NAM.2,NAM.3 ;REPORT ERROR
        003572  104010  001153                                                                  MOV R1,SAVREG
        003574  104301  001147                                                                  MOV NAM.3,R1
        003576  100461                                                                          MOV R1,-(SP)
        003577  104301  001146                                                                  MOV NAM.2,R1
        003601  100461                                                                          MOV R1,-(SP)
        003602  104301  001145                                                                  MOV NAM.1,R1
        003604  100461                                                                          MOV R1,-(SP)
        003605  104301  001153                                                                  MOV SAVREG,R1
        003607  025022                                                                          CALL RERROR     ;ERROR # 2023.
        003610  103747                                                                          .WORD ERHARD+ERRN
        003611  031554                                                                          .WORD <PRMS*10000>+MS2023
```

```
                                                .SBTTL  DIAGNOSE/SET UP RESPONSE TO HOST AND EXIT
                                        ; *** SET UP RESPONSE PACKET
003612  104302  000741          DO.DIO: MOV     SDI,R2                  ;RESTORE PORT INDICATOR
003614  104300  001152  001502          MOV     LUNIT,OUT.01            ;SAVE UNIT NUMBER
003617  115000  001144                  TST     DIAG.1                  ;IF DIAG.1 = 0?
003621  053625                          BNE     1$                      ;IF NOT, SKIP
003622  114000  001503                  CLR     OUT.02                  ;SET UP PROPER OP-CODE VALUE
003624  003630                          BR      2$                      ;CONTINUE
003625  104200  000003  001503  1$:     MOV     #3,OUT.02               ;SET UP REQUEST
003630                          2$:
003630  104300  000744  001504          MOV     ST,OUT.03               ;REGION ID SET
003633  114000  001505                  CLR     OUT.04                  ;
003635  104200  177777  001144          MOV     #177777,DIAG.1          ;MAKE SURE DIAG.1 HAS NONE ZERO VALUE IN IT
                                                                        ; FOR NEXT TIME THROUGH
003640  114002                          CLR     R2
003641  000000                          RETURN
```

```
                                        .SBTTL  READ MEMORY SUBROUTINE
                                ;:+
                                ;       ROUTINE NAME:   RDMEM
                                ;
                                ;       DESCRIPTION:
                                ;       THIS ROUTINE DOES THE XFC READ MEMORY (FROM A SPECIFIED REGION
                                ;       AND OFFSET) OF DATA FROM THE DRIVE.  THIS ROUTINE CALLS THE
                                ;       SEND AND RCV (TALKER) ROUTINE AND JUDGES IF THE SDI COMMAND
                                ;       WAS PROPERLY SENT.
                                ;
                                ;       INPUT:  ALL PARAMETERS FOR THE READ MEMORY COMMAND
                                ;               MUST BE SET.  THAT MEANS REGION ID, THE OFFSET,
                                ;               AND THE BYTE COUNT MUST BE GIVEN THE APPRORIATE VALUES
                                ;               AND STORED IN THE RDM PACKET
                                ;
                                ;       OUTPUT: R2 = 0 MEANS ALL IS OK
                                ;               R2 NOT = 0, READ FAILED (EITHER SEND OR RECIEVE OF THE COMMAND)
                                ;               ST HAS THE READ DATA
                                ;:-
003642                          RDMEM:
003642                                  PUSH    <R0,R1,R3>              ;SAVE THESE REGISTERS
003642  100467                                                                  MOV R0,-(SP)
003643  100461                                                                  MOV R1,-(SP)
003644  100463                                                                  MOV R3,-(SP)
003645  104203  001422                  MOV     #CR.RDM,R3              ;R3 -> RDM PACKET
003647  104302  000741                  MOV     SDI,R2                  ;R2 = PORT
003651  024712                          CALL    TALKER                  ;SEND COMMAND AND RECEIVE RESPONSE
003652  115003                          TST     R3                      ;ERRORS?
003653  013671                          BEQ     2$                      ;IF NONE SO FAR, CONTINUE
003654  110203                          ROL     R3                      ;ERROR ON SEND?
003655  043662                          BCC     1$                      ;IF CARRY CLEAR (RECEIVE ERROR) BRANCH
                                ; *** REPORT TRANSMISSION ERROR
003656                                  ERRHRD  MS2024
003656  025022                                                                  CALL RERROR     ;ERROR # 2024.
003657  103750                                                                  .WORD ERHARD+ERRN
003660  001621                                                                  .WORD <PRMS*10000>+MS2024
003661  003722                          BR      4$                      ;ERROR EXIT
                                ; *** REPORT RECEPTION ERROR
003662  024513                  1$:     CALL    TYPERR                  ;GO CHECK TYPE OF ERROR
003663                                  ERRHRD  MS2025,R0,R3
003663  100463                                                                  MOV R3,-(SP)
003664  100467                                                                  MOV R0,-(SP)
003665  025022                                                                  CALL RERROR     ;ERROR # 2025.
003666  103751                                                                  .WORD ERHARD+ERRN
003667  021653                                                                  .WORD <PRMS*10000>+MS2025
003670  003722                          BR      4$                      ;ERROR EXIT
                                ; *** CHECK RESPONSE OP-CODE FROM DRIVE
003671  106207  000162          2$:     CMP     #RDM.EN,R0              ;CHECK RESPONSE
003673  013720                          BEQ     3$                      ;IF IT MATCHES, OK, EXIT
003674  106207  000175                  CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
003676  053704                          BNE     6$                      ;IF NOT, UNRECOGNIZED RESPONSE
003677  024274                          CALL    STOSTA                  ;CHECK STATUS
003700                                  ERRHRD  MS2026                  ;
003700  025022                                                                  CALL RERROR     ;ERROR # 2026.
003701  103752                                                                  .WORD ERHARD+ERRN
003702  001712                                                                  .WORD <PRMS*10000>+MS2026
003703  003722                          BR      4$
```

READ MEMORY SUBROUTINE

```
     44 003704                          6$:     ERRHRD  MS2027,#RDM.EN,R0
        003704  100467                                                      MOV R0,-(SP)
        003705  104010  001153                                              MOV R1,SAVREG
        003707  104201  000162                                              MOV #RDM.EN,R1
        003711  100461                                                      MOV R1,-(SP)
        003712  104301  001153                                              MOV SAVREG,R1
        003714  025022                                                      CALL RERROR    ;ERROR # 2027.
        003715  103753                                                      .WORD ERHARD+ERRN
        003716  021737                                                      .WORD <PRMS*10000>+MS2027
     45 003717  003722                          BR      4$                 ;ERROR EXIT
     46 003720  114002                  3$:     CLR     R2                 ;ALL OK, EXIT
     47 003721  003724                          BR      5$                 ;
     48 003722  104202  000001          4$:     MOV     #1,R2              ;ERROR OCCURED, SET R2
     49 003724                          5$:     POP     <R3,R1,R0>
        003724  104263                                                                  MOV (SP)+,R3
        003725  104261                                                                  MOV (SP)+,R1
        003726  104267                                                                  MOV (SP)+,R0
     50 003727  000000                          RETURN
```

```
                                        .SBTTL  WRITE MEMORY SUBROUTINE
                                ;:+
                                ;       WRTMEM
                                ;
                                ;       DESCRIPTION:
                                ;       THIS ROUTINE DOES THE XFC WRITE MEMORY TO A SPECIFIED
                                ;       REGION ID AND OFFSET WITH A BYTE COUNT AND DATA.
                                ;       THIS ROUTINE CALLS THE SEND AND RCV (TALKER) ROUTINE
                                ;       AND JUDGES IF THE SDI COMMAND WAS PROPERLY SENT.
                                ;
                                ;       INPUT:  ALL PARAMETERS FOR THE WRITE MEMORY COMMAND MUST BE SET.
                                ;               REGION ID, OFFSET, BYTE COUNT AND THE DATA MUST BE STORED
                                ;               IN THE PACKET.
                                ;
                                ;       OUTPUT: R2 = 0 MEANS WRITE MEMORY COMMAND WAS PROCESSED SUCCESSFULLY
                                ;               R2 NOT = 0, WRITE FAILED (EITHER SEND OR RECEIVE OF THE COMMAND)
                                ;:-
003730  104203  001206        WRTMEM: MOV     #CR.WRM,R3              ;R3->MEM WRITE PACKET
003732  104302  000741                MOV     SDI,R2                  ;R2 = PORT
003734  024712                       CALL    TALKER                  ;SEND COMMAND AND RECEIVE RESPONSE FROM DRIVE
003735  115003                       TST     R3                      ;SEE IF ERROR OCCURRED
003736  013754                       BEQ     2$                      ;IF NO ERRORS, BRANCH
003737  110203                       ROL     R3                      ;SHIFT HIGH BIT INTO CARRY BIT
003740  043745                       BCC     1$                      ;IF CARRY CLEAR (RECEIVE ERROR) BRANCH
                                ; *** REPORT TRANSMISSION ERROR
003741                               ERRHRD  MS2028
003741  025022                                                                 CALL RERROR    ;ERROR # 2028.
003742  103754                                                                 .WORD ERHARD+ERRN
003743  002027                                                                 .WORD <PRMS*10000>+MS2028
003744  004005                       BR      4$
                                ; *** REPORT RECEPTION ERROR
003745  024513                  1$:  CALL    TYPERR                  ;FIND OUT WHAT TYPE OF ERROR OCCURED
003746                               ERRHRD  MS2029,R0,R3
003746  100463                                                                 MOV R3,-(SP)
003747  100467                                                                 MOV R0,-(SP)
003750  025022                                                                 CALL RERROR    ;ERROR # 2029.
003751  103755                                                                 .WORD ERHARD+ERRN
003752  022062                                                                 .WORD <PRMS*10000>+MS2029
003753  004005                       BR      4$
                                ; *** SET UP RESPONSE PACKET
003754  106207  000176          2$:  CMP     #COMPLT,R0              ;DID IT COMPLETE
003756  014003                       BEQ     3$                      ;IF SO CONTINUE
003757  106207  000175               CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
003761  053767                       BNE     6$                      ;IF NOT, UNRECOGNIZED RESPONSE
003762  024274                       CALL    STOSTA                  ;CHECK STATUS
003763                               ERRHRD  MS2030                  ;
003763  025022                                                                 CALL RERROR    ;ERROR # 2030.
003764  103756                                                                 .WORD ERHARD+ERRN
003765  002121                                                                 .WORD <PRMS*10000>+MS2030
003766  004005                       BR      4$
003767                          6$:  ERRHRD  MS2031,#COMPLT,R0
003767  100467                                                                 MOV R0,-(SP)
003770  104010  001153                                                         MOV R1,SAVREG
003772  104201  000176                                                         MOV #COMPLT,R1
003774  100461                                                                 MOV R1,-(SP)
003775  104301  001153                                                         MOV SAVREG,R1
003777  025022                                                                 CALL RERROR    ;ERROR # 2031.
```

```
        004000  103757                                                  .WORD ERHARD+ERRN
        004001  022147                                                  .WORD <PRMS*10000>+MS2031
     41 004002  004005                          BR      4$              ;ERROR EXIT
     42 004003  114002                  3$:     CLR     R2              ;ALL OK, EXIT
     43 004004  004007                          BR      5$              ;
     44 004005  104202  000001          4$:     MOV     #1,R2           ;ERROR OCCURED, SET R2
     45 004007  000000                  5$:     RETURN
```

RUN (OR SPIN UP) COMMAND

```
                                        .SBTTL  RUN (OR SPIN UP) COMMAND
                                ;ISSUE RUN COMMAND AND CHECK THAT IT PERFORMS PROPERLY
                                
004010  104203  001465    RUNDR:  MOV     #CR.RUN,R3              ;R3 -> COMMAND
004012  024712                    CALL    TALKER                  ;SEND TO DRIVE
004013  115003                    TST     R3                      ;ALL OK?
004014  014031                    BEQ     60$                     ;IF SO, CHECK RESPONCE CODE
004015  034022                    BPL     50$                     ;ELSE, IS MSB SET?/IF NOT, RECEIVE ERROR
004016                            ERRHRD  MS2032                  ;REPORT SEND ERROR
004016  025022                                                                    CALL RERROR     ;ERROR # 2032.
004017  103760                                                                    .WORD ERHARD+ERRN
004020  002237                                                                    .WORD <PRMS*10000>+MS2032
004021  004057                    BR      70$                     ;
004022  024513                50$:    CALL    TYPERR
004023                            ERRHRD  MS2033,R0,R3            ;REPORT RECEIVE ERROR
004023  100463                                                                    MOV R3,-(SP)
004024  100467                                                                    MOV R0,-(SP)
004025  025022                                                                    CALL RERROR     ;ERROR # 2033.
004026  103761                                                                    .WORD ERHARD+ERRN
004027  022265                                                                    .WORD <PRMS*10000>+MS2033
004030  004057                    BR      70$                     ;
004031  106207  000176        60$:    CMP     #COMPLT,R0              ;CHECK RESPONSE CODE
004033  014062                    BEQ     75$                     ;IF OK, CONTINUE
004034  106207  000175            CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
004036  054044                    BNE     65$                     ;IF NOT, UNRECOGNIZED RESPONSE
004037  024274                    CALL    STOSTA                  ;CHECK STATUS
004040                            ERRHRD  MS2034                  ;
004040  025022                                                                    CALL RERROR     ;ERROR # 2034.
004041  103762                                                                    .WORD ERHARD+ERRN
004042  002317                                                                    .WORD <PRMS*10000>+MS2034
004043  004057                    BR      70$
004044                        65$:    ERRHRD  MS2035,#COMPLT,R0       ;IF NOT, REPORT ERROR
004044  100467                                                                    MOV R0,-(SP)
004045  104010  001153                                                            MOV R1,SAVREG
004047  104201  000176                                                            MOV #COMPLT,R1
004051  100461                                                                    MOV R1,-(SP)
004052  104301  001153                                                            MOV SAVREG,R1
004054  025022                                                                    CALL RERROR     ;ERROR # 2035.
004055  103763                                                                    .WORD ERHARD+ERRN
004056  022340                                                                    .WORD <PRMS*10000>+MS2035
004057  104202  000001        70$:    MOV     #1,R2                   ;R2 IS NOT ZERO MEANING ERROR
004061  004063                    BR      80$
004062  114002                75$:    CLR     R2                      ;R2 IS ZERO, NOT ERROR
004063  000000                80$:    RETURN
```

```
                                        .SBTTL  RECALIBRATE COMMAND
                                ;ISSUE RECALIBRATE COMMAND AND CHECK THAT IT PERFORMS PROPERLY
                                ;
004064  104203  001473      RECAL:  MOV     #CR.INR,R3              ;R3 -> COMMAND
004066  024712                      CALL    TALKER                  ;SEND TO DRIVE
004067  115003                      TST     R3                      ;ALL OK?
004070  014105                      BEQ     90$                     ;IF SO, CHECK RESPONCE CODE
004071  034076                      BPL     80$                     ;ELSE, IS MSB SET?/IF NOT, RECEIVE ERROR
004072                              ERRHRD  MS2036                  ;REPORT SEND ERROR
004072  025022                                                                  CALL RERROR     ;ERROR # 2036.
004073  103764                                                                  .WORD ERHARD+ERRN
004074  002424                                                                  .WORD <PRMS*10000>+MS2036
004075  004133                      BR      100$                    ;
004076  024513                80$:  CALL    TYPERR
004077                              ERRHRD  MS2037,R0,R3            ;REPORT RECEIVE ERROR
004077  100463                                                                  MOV R3,-(SP)
004100  100467                                                                  MOV R0,-(SP)
004101  025022                                                                  CALL RERROR     ;ERROR # 2037.
004102  103765                                                                  .WORD ERHARD+ERRN
004103  022456                                                                  .WORD <PRMS*10000>+MS2037
004104  004133                      BR      100$                    ;
004105  106207  000176        90$:  CMP     #COMPLT,R0              ;CHECK RESPONSE CODE
004107  014136                      BEQ     105$                    ;IF OK, CONTINUE
004110  106207  000175              CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
004112  054120                      BNE     95$                     ;IF NOT, UNRECOGNIZED RESPONSE
004113  024274                      CALL    STOSTA                  ;CHECK STATUS
004114                              ERRHRD  MS2038                  ;
004114  025022                                                                  CALL RERROR     ;ERROR # 2038.
004115  103766                                                                  .WORD ERHARD+ERRN
004116  002514                                                                  .WORD <PRMS*10000>+MS2038
004117  004133                      BR      100$
004120                          95$:  ERRHRD  MS2039,#COMPLT,R0       ;IF NOT, REPORT ERROR
004120  100467                                                                  MOV R0,-(SP)
004121  104010  001153                                                          MOV R1,SAVREG
004123  104201  000176                                                          MOV #COMPLT,R1
004125  100461                                                                  MOV R1,-(SP)
004126  104301  001153                                                          MOV SAVREG,R1
004130  025022                                                                  CALL RERROR     ;ERROR # 2039.
004131  103767                                                                  .WORD ERHARD+ERRN
004132  022541                                                                  .WORD <PRMS*10000>+MS2039
004133  104202  000001        100$: MOV     #1,R2
004135  004137                      BR      110$
004136  114002                105$: CLR     R2
004137  000000                110$: RETURN
```

```
                                        .SBTTL  GET STATUS SUBROUTINE
                                ;
                                ;       GET STATUS
                                ;
                                ;       OUTPUT  R3 = 0 IF OK
                                ;               R3 = 1 IF ERROR
                                ;               R4 USED AS A DECREMENT COUNTER
                                ;
004140  114004          GTSTAT: CLR     R4
004141  104203  001171  3$:     MOV     #CR.GST,R3              ;POINT TO GET STATUS COMMAND
004143  024712                  CALL    TALKER                  ; SEND AND RECEIVE COMMAND
004144  115003                  TST     R3                      ; SEE IF ERROR OCCURED
004145  014165                  BEQ     2$                      ; IF NOT, BRANCH
004146  110203                  ROL     R3                      ; SEE IF RECEIVE ERROR
004147  044156                  BCC     1$                      ; IF SO, BRANCH
004150  117404                  DEC     R4                      ; IF SEND ERROR, COUNTER DONE?
004151  054141                  BNE     3$                      ; IF NOT, TRY AGAIN
004152                          ERRHRD  MS2040
004152  025022                                                                  CALL RERROR     ;ERROR # 2040.
004153  103770                                                                  .WORD ERHARD+ERRN
004154  002631                                                                  .WORD <PRMS*10000>+MS2040
004155  004213                  BR      ER.END
004156  024513          1$:     CALL    TYPERR
004157                          ERRHRD  MS2041,R0,R3
004157  100463                                                                  MOV R3,-(SP)
004160  100467                                                                  MOV R0,-(SP)
004161  025022                                                                  CALL RERROR     ;ERROR # 2041.
004162  103771                                                                  .WORD ERHARD+ERRN
004163  022663                                                                  .WORD <PRMS*10000>+MS2041
004164  004213                  BR      ER.END
                                ; *** LOOK AT RESPONSE OP-CODE FROM DRIVE -- SHOULD BE DATA RESPONSE
004165  106207  000366  2$:     CMP     #STSRES,R0              ;CHECK OP-CODE
004167  014216                  BEQ     OK.END                  ;BRANCH IF A MATCH
004170  106207  000175          CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
004172  054200                  BNE     4$                      ;IF NOT, UNRECOGNIZED RESPONSE
004173  024274                  CALL    STOSTA
004174                          ERRHRD  MS2042                  ;
004174  025022                                                                  CALL RERROR     ;ERROR # 2042.
004175  103772                                                                  .WORD ERHARD+ERRN
004176  002721                                                                  .WORD <PRMS*10000>+MS2042
004177  004213                  BR      ER.END
004200                  4$:     ERRHRD  MS2043,#STSRES,R0       ;WRONG RESPONSE FROM DRIVE
004200  100467                                                                  MOV R0,-(SP)
004201  104010  001153                                                          MOV R1,SAVREG
004203  104201  000366                                                          MOV #STSRES,R1
004205  100461                                                                  MOV R1,-(SP)
004206  104301  001153                                                          MOV SAVREG,R1
004210  025022                                                                  CALL RERROR     ;ERROR # 2043.
004211  103773                                                                  .WORD ERHARD+ERRN
004212  022746                                                                  .WORD <PRMS*10000>+MS2043
004213  104203  000001  ER.END: MOV     #1,R3                   ;R3 = NONE ZERO VALUE WHEN DONE(ERROR)
004215  004217                  BR      EX.END                  ;EXIT
004216  114003          OK.END: CLR     R3                      ;R3 = 0 WHEN DONE (NO ERROR)
004217  000000          EX.END: RETURN
```

```
                                        .SBTTL  CLEAR DRIVE SUBROUTINE
                                ;
                                ;       DRIVE CLEAR
                                ;
                                ;       OUTPUT  R3 = 0 IF OK
                                ;               R3 = 1 IF ERROR
                                ;               R4 USED AS A DECREMENT COUNTER
                                ;
004220  114004                  CLRDRV: CLR     R4                      ; CLEAR DECEMENT COUNTER
004221  104203  001177          3$:     MOV     #CR.DRC,R3              ; POINT TO DRIVE CLEAR COMMAND
004223  024712                          CALL    TALKER                  ; SEND AND RECEIVE COMMAND
004224  115003                          TST     R3                      ; SEE IF ERROR OCCURED
004225  014245                          BEQ     2$                      ; IF NOT, BRANCH
004226  110203                          ROL     R3                      ; SEE IF RECEIVE ERROR
004227  044236                          BCC     1$                      ; IF SO, BRANCH
004230  117404                          DEC     R4                      ; ON SEND ERROR, IS COUNTER DONE?
004231  054221                          BNE     3$                      ; IF NOT, TRY AGAIN
004232                                  ERRHRD  MS2044
004232  025022                                                                  CALL RERROR     ;ERROR # 2044.
004233  103774                                                                  .WORD ERHARD+ERRN
004234  003035                                                                  .WORD <PRMS*10000>+MS2044
004235  004213                          BR      ER.END
004236  024513                  1$:     CALL    TYPERR
004237                                  ERRHRD  MS2045,R0,R3
004237  100463                                                                  MOV R3,-(SP)
004240  100467                                                                  MOV R0,-(SP)
004241  025022                                                                  CALL RERROR     ;ERROR # 2045.
004242  103775                                                                  .WORD ERHARD+ERRN
004243  023067                                                                  .WORD <PRMS*10000>+MS2045
004244  004213                          BR      ER.END
                                ; *** CHECK RESPONSE CODE
004245  106207  000176          2$:     CMP     #COMPLT,R0              ; SEE IF CORRECT RESPONSE CODE RECEIVED
004247  014216                          BEQ     OK.END                  ; IF SUCCESSFUL, BRANCH
004250  106207  000175                  CMP     #UNSSUC,R0              ;IF NOT CORRECT RESPONSE, UNSSUC?
004252  054260                          BNE     4$                      ;IF NOT, UNRECOGNIZED RESPONSE
004253  024274                          CALL    STOSTA                  ;CHECK STATUS
004254                                  ERRHRD  MS2046                  ;
004254  025022                                                                  CALL RERROR     ;ERROR # 2046.
004255  103776                                                                  .WORD ERHARD+ERRN
004256  003125                                                                  .WORD <PRMS*10000>+MS2046
004257  004213                          BR      ER.END
004260                          4$:     ERRHRD  MS2047,#COMPLT,R0       ;CLEAR ERROR FAILED
004260  100467                                                                  MOV R0,-(SP)
004261  104010  001153                                                          MOV R1,SAVREG
004263  104201  000176                                                          MOV #COMPLT,R1
004265  100461                                                                  MOV R1,-(SP)
004266  104301  001153                                                          MOV SAVREG,R1
004270  025022                                                                  CALL RERROR     ;ERROR # 2047.
004271  103777                                                                  .WORD ERHARD+ERRN
004272  023152                                                                  .WORD <PRMS*10000>+MS2047
004273  004213                          BR      ER.END
```

```
                                        .SBTTL  STORE STATUS ROUTINE
                                ;       STORE STATUS
                                ;               MOVE WHAT DATA IS IN ST AND PUT IT IN OUTPUT BUFFER
004274                          STOSTA:
004274                                  PUSH    <R0,R1,R2>
004274  100467                                                                          MOV R0,-(SP)
004275  100461                                                                          MOV R1,-(SP)
004276  100462                                                                          MOV R2,-(SP)
004277  104207  001506                 MOV     #OUT.05,R0              ;R0 -> OUTPUT BUFFER
004301  104201  004252                 MOV     #STAT0,R1               ;MOVE FORMAT ADDRESS IN OUTPUT BUFFER
004303  100271                         MOV     R1,(R0)+
004304  104302  000741                 MOV     SDI,R2                  ; R2 PORT INDICATOR
004306  060007                         XFC     STATUS                  ; GET STATUS
004307  103201  077270                 BIC     #^C100507,R1            ; CLEAR UNUSED BITS
004311  100271                         MOV     R1,(R0)+                ; STORE
004312  104201  000753                 MOV     #ST+7,R1                ;R1 -> INPUT BUFFER
004314  104412                 1$:     MOV     -(R1),R2                ;SAVE
004315  100272                         MOV     R2,(R0)+
004316  106201  000744                 CMP     #ST,R1                  ;DONE?
004320  054314                         BNE     1$                      ;IF NOT, CONTINUE
004321                                  POP     <R2,R1,R0>
004321  104262                                                                          MOV (SP)+,R2
004322  104261                                                                          MOV (SP)+,R1
004323  104267                                                                          MOV (SP)+,R0
004324  000000                         RETURN
```

```
                                .SBTTL  CONVERT MEMORY -- SKEWED BY BYTE

                        ;:+
                        ;       ROUTINE NAME:   CONMEM
                        ;
                        ;       DESCRIPTION:
                        ;       MEMORY IS SENT TO THE DM ROUTINE STARTING ON THE ODD BYTE BOUNDARY.
                        ;       THIS ROUTINE SHIFTS THE DATA TO START ON THE EVEN BYTE BOUNDARY AND
                        ;       STORES EACH WORD INTO A SPECIFIED BUFFER AREA.  THE DATA IS STORED AS
                        ;       FOLLOWS:
                        ;               ST:     DATA BYTE 0     BYTE COUNT
                        ;                       DATA BYTE 2     DATA BYTE 1
                        ;                       DATA BYTE 4     DATA BYTE 3
                        ;                       .....           DATA BYTE 5
                        ;
                        ;       AFTER EXECUTION, THD DATA IS STORED LIKE THIS:
                        ;               OUT.XX: DATA BYTE 1     DATA BYTE 0
                        ;                       DATA BYTE 3     DATA BYTE 2
                        ;                       DATA BYTE 5     DATA BYTE 4
                        ;                       .....           DATA BYTE 6
                        ;
                        ;       INPUT:  R4 -> BUFFER AREA'S FIRST LOCATION
                        ;               ST HAS INPUT DATA
                        ;
                        ;       OUTPUT: OUTPUT BUFFER HAS SHIFTED DATA
                        ;
                        ;:-
004325                  CONMEM: PUSH    <R0,R1,R2,R3,R4>
004325  100467                                                          MOV R0,-(SP)
004326  100461                                                          MOV R1,-(SP)
004327  100462                                                          MOV R2,-(SP)
004330  100463                                                          MOV R3,-(SP)
004331  100464                                                          MOV R4,-(SP)
004332  104203  000744                 MOV     #ST,R3                  ;R3 -> BYTE COUNT
004334  104137                          MOV     (R3),R0                 ;R0 = BYTE COUNT
004335  103207  177400                 BIC     #HIBYTE,R0              ;STRIP OFF EXTRANEOUS
004337  104231                  1$:     MOV     (R3)+,R1                ;R1 = EVEN BYTE OF DATA
004340  103201  000377                 BIC     #LOBYTE,R1              ;STRIP OFF EXTRANEOUS
004342  117407                          DEC     R0                      ;DONE?
004343  014355                          BEQ     2$                      ;IF YES, GO STORE LAST BYTE
004344  104132                          MOV     (R3),R2                 ;R2 = ODD BYTE
004345  103202  177400                 BIC     #HIBYTE,R2              ;STRIP OFF EXTRANEOUS
004347  101021                          BIS     R2,R1                   ;R1 HAS WHOLE WORD(BYTES REVERSED)
004350  110701                          SWAB    R1                      ;SWITCH BYTES
004351  100241                          MOV     R1,(R4)+                ;STORE DATA
004352  117407                          DEC     R0                      ;DONE?
004353  014357                          BEQ     3$                      ;IF SO, EXIT(EVEN # OF BYTES)
004354  004337                          BR      1$                      ;ELSE, CONTINUE
004355  110701                  2$:     SWAB    R1                      ;ODD # BYTES TO GET HERE
004356  100141                          MOV     R1,(R4)                 ;STORE LAST BYTE
004357                          3$:     POP     <R4,R3,R2,R1,R0>
004357  104264                                                          MOV (SP)+,R4
004360  104263                                                          MOV (SP)+,R3
004361  104262                                                          MOV (SP)+,R2
004362  104261                                                          MOV (SP)+,R1
004363  104267                                                          MOV (SP)+,R0
004364  000000                          RETURN
```

```
                                .SBTTL  GET BYTE COUNT
                        ;:+
                        ;       ROUTINE NAME:   GETBCN
                        ;
                        ;       DESCRIPTION:
                        ;       DATA IS READ BY THE DM PROGRAM AFTER THE DIAGNOSE COMMAND.
                        ;       WITHIN THIS DATA, THE DRIVE SPECIFIES IF MORE DATA NEEDS TO
                        ;       BE READ.  IF IT IS, THE BYTE COUNT MUST BE CONVERTED FROM
                        ;       ITS PRESENT FORM WHICH FOLLOWS THIS FORMAT:
                        ;
                        ;       BIT 15                                          BIT 0
                        ;       +-----------------------+-----------------------+
                        ;OUT.08:!       ASCII COUNT     !     BINARY COUNT      !
                        ;       +-----------------------+-----------------------+
                        ;
                        ;       THE TOTAL BYTE COUNT IS FOUND BY THIS FORMULA:
                        ;                       TBC = AC + (4*BC)
                        ;       OR THE TOTAL BYTE COUNT EQUALS THE ASCII COUNT PLUS FOUR
                        ;       TIMES THE BINARY COUNT.  THE BINARY COUNT IS THE NUMBER OF
                        ;       32 BIT BINARY VALUES INCLUDED IN THE REPORT.  THE ASCII
                        ;       COUNT IS THE NUMBER OF ASCII CHARACTERS INCLUDED IN THE
                        ;       REPORT.
                        ;
                        ;       INPUT:  OUT.08 = ASCII COUNT AND BINARY COUNT IN THE FORM STATED ABOVE
                        ;
                        ;       OUTPUT: RDM+3 HAS UPDATED BYTE COUNT.
                        ;
                        ;:-
                        ;
004365                  GETBCN: PUSH    <R0,R1,R2,R3,R4,R5>
004365  100467                                                          MOV R0,-(SP)
004366  100461                                                          MOV R1,-(SP)
004367  100462                                                          MOV R2,-(SP)
004370  100463                                                          MOV R3,-(SP)
004371  100464                                                          MOV R4,-(SP)
004372  100465                                                          MOV R5,-(SP)
004373  114007                  CLR     R0
004374  110207                  ROL     R0                      ;CLEAR CARRY
004375  104307  001511          MOV     OUT.08,R0               ;GET BYTE COUNT
004377  103207  177400          BIC     #HIBYTE,R0              ;R0 = BINARY COUNT= # 32 BIT WORDS
004401  110207                  ROL     R0                      ;
004402  110207                  ROL     R0                      ;R0 = BINARY COUNT IN BYTES
004403  104301  001511          MOV     OUT.08,R1               ;R1 = BYTE COUNT
004405  110701                  SWAB    R1                      ;ASCII COUNT IN LO BYTE
004406  103201  177400          BIC     #HIBYTE,R1              ;R1 = ASCII COUNT
004410  104012                  MOV     R1,R2                   ;SAVE IN R2 = ASCII COUNT
004411  105017                  ADD     R1,R0                   ;R0 = TOTAL BYTE COUNT
                        ; *** IS SIZE OF THE PACKET TOO BIG FOR BUFFER AVAILABLE?
004412  104201  000052          MOV     #42.,R1                 ;42. IS THE MAXIMUM # OF BYTES LEFT
004414  106017                  CMP     R1,R0                   ;IS TOTAL BYTE COUNT > 42.?
004415  034502                  BPL     5$                      ;IF NOT, GO STORE TOTAL BYTE COUNT
                        ; *** SAVE OUT.RQ DATA
004416  104204  001502          MOV     #OUT.01,R4              ;R4-> OUT.RQ DATA
004420  104205  001104          MOV     #ST+140,R5              ;R5->SAVE AREA
004422  104241          1$:     MOV     (R4)+,R1                ;SAVE
004423  100251                  MOV     R1,(R5)+                ;
004424  106204  001543          CMP     #OUT.34,R4              ;DONE?
004426  054422                  BNE     1$                      ;IF NOT, CONTINUE
```

```
                                ;
                                        MSSGE   MSG2
004427                                                                          MOV     LUNIT,OUT.01
004427  104300  001152  001502                                                  MOV     #MSG2,OUT.02
004432  104200  003342  001503                                                          MOV R0,-(SP)
004435  100467                                                                          MOV R1,-(SP)
004436  100461
004437  104207  060015                                                          MOV     #MESSAG,R0
004441  024652                                                                  CALL    HOSTRQ
004442  104261                                                                          MOV (SP)+,R1
004443  104267                                                                          MOV (SP)+,R0
                                ; *** RESTORE OUT.RQ DATA
004444  104204  001502                  MOV     #OUT.01,R4              ;R4-> OUT.RQ DATA
004446  104205  001104                  MOV     #ST+140,R5              ;R5->SAVE AREA
004450  104251          2$:     MOV     (R5)+,R1                ;RESTORE
004451  100241                  MOV     R1,(R4)+                ;
004452  106204  001543                  CMP     #OUT.34,R4              ;DONE?
004454  054450                  BNE     2$                      ;IF NOT, CONTINUE
                                ;
004455  104073                  MOV     R0,R3                   ;SAVE ASCII COUNT + 4*BINARY COUNT IN R3
004456  104207  000052                  MOV     #42.,R0                 ;ELSE, MAXIMUM BYTE COUNT IS STORED.
                                ; *** ADJUST ASCII COUNT IF TOO LARGE
004460  107023                  SUB     R2,R3                   ;R3 = BINARY COUNT
004461  106073                  CMP     R0,R3                   ;IF BINARY COUNT ALONE > 42.?
004462  074471                  BMI     3$                      ;IF SO, TO CLEAR ASCII COUNT AND ADJUST BINARY COUNT
                                ; *** ASCII COUNT IS > 42. TO GET HERE
004463  105032                  ADD     R3,R2                   ;R2 = TOTAL BYTE COUNT
004464  107072                  SUB     R0,R2                   ;R2 = ASCII COUNT ADJUSTMENT
004465  110702                  SWAB    R2                      ;PUT ASCII COUNT ADJUSTMENT IN UPPER BYTE
004466  107020  001511                  SUB     R2,OUT.08               ;SUBTRACT FROM OUT.08 FOR HOST
004470  004502                  BR      5$                      ;EXIT
                                ; *** ADJUST BINARY COUNT IF TOO LARGE/CLEAR OUT ASCII COUNT
004471  103200  177400  001511  3$:     BIC     #HIBYTE,OUT.08          ;CLEAR HI BYTE OF OUT.08
004474  107073                  SUB     R0,R3                   ;R3 = BINARY COUNT ALONE
004475  117400  001511          4$:     DEC     OUT.08                  ;ADJUST BINARY COUNT
004477  107203  000004                  SUB     #4,R3
004501  034475                  BPL     4$
                                ; *** STORE IN READ MEMORY PACKET AND EXIT
004502                          5$:
004502  104070  001432                  MOV     R0,RDM+3                ;STORE IN READ MEMORY PACKET
004504                          POP     <R5,R4,R3,R2,R1,R0>
004504  104265                                                                          MOV (SP)+,R5
004505  104264                                                                          MOV (SP)+,R4
004506  104263                                                                          MOV (SP)+,R3
004507  104262                                                                          MOV (SP)+,R2
004510  104261                                                                          MOV (SP)+,R1
004511  104267                                                                          MOV (SP)+,R0
004512  000000                  RETURN
```

```
                                                .SBTTL  TYPE WHAT KIND OF RECEIVE ERROR
                                        ;:+
                                        ;       TYPE ERROR
                                        ;
                                        ;       DESCRIPTION:
                                        ;       THIS ROUTINE PRINTS A REPORT TO TELL WHAT KIND OF ERROR OCCURED IF A
                                        ;       RECEIVE XFC DID NOT SUCCESSFULLY COMPLETE.
                                        ;
                                        ;       INPUT:  R3 = ERROR VALUE FROM THE XFC SAVED IN TALKER ROUTINE SHIFTED LEFT ONCE
                                        ;                       IF = 1 TIMEOUT
                                        ;                       IF = 2 1ST WORD WAS NOT A START FRAME
                                        ;                       IF = 4 FRAMING ERROR ON SDI LEVEL 0 READ
                                        ;                       IF = 10 CHECKSUM ERROR ON SDI LEVEL 0 READ
                                        ;                       IF = 20 BUFFER SIZE WAS SMALLER THAN RESPONSE
                                        ;:-
004513  110603                  TYPERR: ROR     R3                      ;READJUST R3
004514  114001                          CLR     R1
004515  110201                          ROL     R1                      ;CLEAR CARRY
004516  104031                          MOV     R3,R1                   ;STORE IN R1
004517  110601                          ROR     R1                      ;ERROR?
004520  044524                          BCC     1$                      ;IF NOT, CONTINUE
004521  104207  003374                  MOV     #MSR0,R0                ;TIMEOUT
004523  004552                          BR      6$                      ;EXIT
004524  110601                  1$:     ROR     R1                      ;ERROR?
004525  044531                          BCC     2$                      ;IF NOT, CONTINUE
004526  104207  003423                  MOV     #MSR1,R0                ;1ST WORD WAS NOT A START FRAME
004530  004552                          BR      6$                      ;EXIT
004531  110601                  2$:     ROR     R1                      ;ERROR?
004532  044536                          BCC     3$                      ;IF NOT, CONTINUE
004533  104207  003453                  MOV     #MSR2,R0                ;FRAMING ERROR ON SDI LEVEL 0 READ
004535  004552                          BR      6$                      ;EXIT
004536  110601                  3$:     ROR     R1                      ;ERROR?
004537  044543                          BCC     4$                      ;IF NOT, CONTINUE
004540  104207  003514                  MOV     #MSR3,R0                ;CHECKSUM ERROR ON SDI LEVEL 0 READ
004542  004552                          BR      6$                      ;EXIT
004543  110601                  4$:     ROR     R1                      ;ERROR?
004544  044550                          BCC     5$                      ;IF NOT, CONTINUE
004545  104207  003555                  MOV     #MSR4,R0                ;BUFFER SIZE SMALLER THAN RESPONSE
004547  004552                          BR      6$                      ;EXIT
004550  104207  003612          5$:     MOV     #MSR5,R0                ;ERROR WASN'T PROPERLY SPECIFIED
004552  000000                  6$:     RETURN
```

```
                                        .SBTTL  DIVIDE BY OCTAL 50 AND FIND ASCII EQUIVALENT
                                ;:+
                                ;       DIVIDE BY 50
                                ;
                                ;       DESCRIPTION:    RAD 50 VALUE IN R4 IS DIVIDED BY 50(OCTAL)
                                ;                       TO STRIP OFF A SINGLE CHARACTER INTO R0.
                                ;                       JUMP TO FNDASC TO FIND THE ASCII EQUIVALENT
                                ;
                                ;       INPUT:  R4 HAS RAD50 VALUE
                                ;
                                ;       OUTPUT: R0 HAS ASCII CHARACTER
                                ;:-
004553                          DIV50:  PUSH    <R1>
004553  100461                                                                  MOV R1,-(SP)
004554  114001                          CLR     R1                      ;R1 = QUO
004555  115401                  1$:     INC     R1                      ;KEEP TRACK OF # OF LOOPS
004556  107204  000050                  SUB     #50,R4                  ;DONE?
004560  034555                          BPL     1$                      ;IF NOT, LOOP
004561  105204  000050                  ADD     #50,R4                  ;R4 = REMAINDER
004563  104047                          MOV     R4,R0                   ;STORE REMAINDER IN R0
004564  117401                          DEC     R1                      ;R1 HAS 1 MORE THAN IT SHOULD
004565  104014                          MOV     R1,R4                   ;QUO IN R4
004566                                  POP     <R1>
004566  104261                                                                  MOV (SP)+,R1
                                ;:+
                                ;       FIND ASCII
                                ;
                                ;       INPUT:  R0 HAS RAD 50 CHARACTER
                                ;:-
004567  115007                  FNDASC: TST     R0                      ;IS = 0?
004570  054574                          BNE     2$                      ;IF NOT, CONTINUE
004571  104207  000040                  MOV     #40,R0                  ;IF SO, SPACE SET
004573  000000                          RETURN
004574  106207  000032          2$:     CMP     #32,R0                  ; > LARGEST LETTER
004576  074602                          BMI     3$                      ;IF SO, CONTINUE
004577  105207  000100                  ADD     #100,R0                 ;SET ASCII LETTER
004601  000000                          RETURN
004602  106207  000033          3$:     CMP     #33,R0                  ; = 33?
004604  054610                          BNE     4$                      ;IF NOT, CONTINUE
004605  104207  000044                  MOV     #44,R0                  ;SET '$'
004607  000000                          RETURN
004610  106207  000034          4$:     CMP     #34,R0                  ; = 34?
004612  054616                          BNE     5$                      ;IF NOT, CONTINUE
004613  104207  000056                  MOV     #56,R0                  ;SET '.'
004615  000000                          RETURN
004616  105207  000022          5$:     ADD     #22,R0                  ;ELSE ADD 22(OCTAL)FOR NUMERAL CHARACTER
004620  000000                          RETURN
```

```
RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE

                                        .SBTTL  RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
004621                                  RDSTAT:
                                        ;
                                        ;RETURN DRIVE STATUS
                                        ;STATUS RETURNED IN DM REGISTER 1
                                        ;
004621                                          PUSH    <R3,R0>                 ; SAVE R3 AND R0
004621  100463                                                                          MOV R3,-(SP)
004622  100467                                                                          MOV R0,-(SP)
004623  104203  000003                         MOV     #3,R3                   ; ALLOW ONLY 3 ERRORS
004625  060007                  STATLP: XFC     STATUS                  ; GET DRIVE'S STATUS
004626  103201  014000                  BIC     #14000,R1               ; CLEAR ERROR PASSING BITS
004630  102201  000400                  BIT     #XMTERR,R1              ; CHECK XMIT ERRORS
004632  014640                          BEQ     STATOK                  ; IF NO ERRORS, BRANCH
004633  117403                          DEC     R3                      ; DECREMENT TRANSMIT ERROR COUNT
004634  054625                          BNE     STATLP                  ; IF ERROR COUNT INCOMPLETE, BRANCH
004635  104201  010000                  MOV     #10000,R1               ; FLAG AS TRANSMIT ERROR
004637  004647                          BR      STATEX                  ; BRANCH
004640  102201  000004          STATOK: BIT     #RCVERR,R1              ; RECIEVER ERRORS
004642  054647                          BNE     STATEX                  ; IF VALID, BRANCH
004643  117403                          DEC     R3                      ; DECREMENT ERROR COUNT
004644  054625                          BNE     STATLP                  ; IF ERROR COUNT NON-ZERO, BRANCH
004645  104201  014000                  MOV     #14000,R1               ; FLAG AS INVALID STATUS ERROR
004647                          STATEX: POP     <R0,R3>                 ; RESTORE R0, R3
004647  104267                                                                          MOV (SP)+,R0
004650  104263                                                                          MOV (SP)+,R3
004651  000000                                  RETURN                          ; RETURN TO CALLING MODULE
```

```
                                .SBTTL  HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
004652                          HOSTRQ:
                                ;
                                ;SEND REQUEST BUFFER TO HOST AND WAIT FOR RESPONSE.
                                ;CLEAR ARGUMENT AREA OF OUT BUFFER IN PREPARATION
                                ;FOR NEXT HOSTRQ CALL.
                                ;
                                ;INPUTS:
                                ;       R0 - HOST REQUEST NUMBER
                                ;       OUT BUFFER LOADED WITH DATA
                                
004652                          PUSH    <R0,R1,R2>
004652  100467                                                                  MOV R0,-(SP)
004653  100461                                                                  MOV R1,-(SP)
004654  100462                                                                  MOV R2,-(SP)
004655  104070  001501          MOV     R0,OUT.RQ               ; STORE REQUEST NUMBER IN BUFFER
004657  104207  001501  SNDAGN: MOV     #OUT.RQ,R0              ; SEND BUFFER TO HOST
004661  104201  000043          MOV     #BUFSIZ,R1
004663  060016                  XFC     MRD
004664  115001                  TST     R1                      ; CHECK FOR ERROR
004665  054657                  BNE     SNDAGN                  ; IF ERROR, TRY AGAIN
004666  104207  001544          MOV     #IN.RQ,R0               ; WAIT FOR RESPONSE FROM HOST
004670  104201  000043          MOV     #BUFSIZ,R1
004672  060017                  XFC     MWR
004673  104200  177777  001501  MOV     #-1,OUT.RQ              ; MAKE REQUEST ILLEGAL
004676  104207  001502          MOV     #OUT.01,R0              ; CLEAR ARGUMENT WORDS IN BUFFER
004700  104201  000042          MOV     #BUFSIZ-1,R1            ; CLEAR ENTIRE BUFFER
004702  114002                  CLR     R2
004703  100272  CLRBUF: MOV     R2,(R0)+
004704  117401          DEC     R1
004705  034703          BPL     CLRBUF
004706                  POP     <R2,R1,R0>
004706  104262                                                                  MOV (SP)+,R2
004707  104261                                                                  MOV (SP)+,R1
004710  104267                                                                  MOV (SP)+,R0
004711  000000          RETURN
```

```
                                        ;
                                        ;       TALKER SENDS AND RECEIVES AN SDI INTERCHANGE
                                        ;
                                        ;INPUTS:
                                        ;       R2 - SDI INTERCONNECT
                                        ;       R3 - POINTER TO COMMAND TABLE CONTAINING APPROPRIATE COMMAND
                                        ;       SDILTO/SDISTO SDI LONG AND SHORT TIMEOUTS, RESPECTIVELY
                                        ;OUTPUTS:
                                        ;       R0 - RETURN OP CODE FROM UNIT
                                        ;       R3 - ERROR CODE - 0 = NO ERROR, 1 = RECEIVE ERROR, 100001 = SEND ERROR
                                        ;
004712                                  TALKER: PUSH    <R1,R4>         ; SAVE REGISTERS
004712  100461                                                                  MOV R1,-(SP)
004713  100464                                                                  MOV R4,-(SP)
004714  104237                                  MOV     (R3)+,R0        ; SET ADR OF SDI COMMAND BUFFER
004715  104231                                  MOV     (R3)+,R1        ; SET BUFFER LENGTH
004716  060004                                  XFC     SEND            ; SEND COMMAND
004717  115001                                  TST     R1              ; DID UNIT ACCEPT COMMAND
004720  014724                                  BEQ     TALK1A          ; IF SO, BRANCH
004721  104203  100001                         MOV     #100001,R3      ; FLAG AS SEND ERROR
004723  004772                                  BR      TALK2B          ; BRANCH TO EXIT
004724  106203  001455                 TALK1A: CMP     #LONG,R3        ; SEE IF LONG TIMEOUT TO BE USED
004726  074732                                  BMI     1$              ; IF SO, BRANCH
004727  104304  001154                         MOV     SDISTO,R4       ; SET UP SHORT TIMEOUT
004731  004734                                  BR      TALK1B          ; BRANCH
004732  104304  001155                 1$:     MOV     SDILTO,R4       ; SET UP LONG TIMEOUT
004734                                  TALK1B:
004734  115000  001151                         TST     SENDHR          ; DO WE SEND HOSTRQ?
004736  014751                                  BEQ     1$              ; IN NOT, CONTINUE
004737  104300  001152  001502                 MOV     LUNIT,OUT.01    ; SEND UNIT NUMBER
004742  114000  001503                         CLR     OUT.02          ; NO MEGABYTES READ
004744  114000  001504                         CLR     OUT.03          ; NO MEGABYTES WRITTEN
004746  104207  060011                         MOV     #T4MXFR,R0      ; SEND SOMETHING TO HOST
004750  024652                                  CALL    HOSTRQ          ;  SO HOST WON'T TIMEOUT DM PROG
004751  104137                          1$:     MOV     (R3),R0         ; SET DATA BUFFER ADDRESS
004752  104631  000001                         MOV     1(R3),R1        ; SET BUFFFR LENGTH
004754  060005                                  XFC     RCV             ; SEND RECEIVE SDI COMMAND
004755  115001                                  TST     R1              ; DID ERROR OCCUR
004756  014771                                  BEQ     TALK2A          ; IF NOT, BRANCH
004757  106201  000001                         CMP     #1,R1           ; SEE IF TIMEOUT
004761  014764                                  BEQ     2$              ; IF SO, BRANCH
004762  104013                                  MOV     R1,R3           ; MOVE ERROR TYPE TO R3 FOR REPORTING
004763  004772                                  BR      TALK2B          ; EXIT
004764  117404                          2$:     DEC     R4              ; DECREMENT TIMEOUT VALUE
004765  054734                                  BNE     TALK1B          ; IF NOT TIMEOUT, BRANCH
004766  104203  000001                         MOV     #1,R3           ; FLAG AS RECIEVE ERROR
004770  004772                                  BR      TALK2B          ; BRANCH TO EXIT
004771  114003                          TALK2A: CLR     R3              ; FLAG AS NO ERRORS
004772                                  TALK2B: POP     <R4,R1>         ; RESTORE R4, R1
004772  104264                                                                  MOV (SP)+,R4
004773  104261                                                                  MOV (SP)+,R1
004774  000000                                  RETURN
```

```
                                ;
                                ;XOR
                                ;
                                ;PERFORM XOR LOGIC FUNCTION ON TWO REGISTERS
                                ;INPUTS:
                                ;       R1, R2 - DATA TO BE XOR'ED
                                ;OUTPUTS:
                                ;       R1 - UNCHANGED
                                ;       R2 - XOR OF TWO INPUTS
                                ;
004775                          XOR:    PUSH R3                         ;SAVE R3
004775  100463                                                                  MOV R3,-(SP)
004776  104013                          MOV R1,R3
004777  103023                          BIC R2,R3
005000  103012                          BIC R1,R2
005001  101032                          BIS R3,R2
005002                          POP R3                                  ;RESTORE R3
005002  104263                                                                  MOV (SP)+,R3
005003  115002                          TST R2                          ;SET CONDITION CODES
005004  000000                          RETURN
```

```
                                        TO:
                                        ;
                                        ;       CALCULATE THE TIMEOUT IN 9SEC INTERVALS (SDI RECEIVE XFC TAKES
                                        ;       9 SEC)
                                        ;
 1 005005                               TO:
 2
 3
 4
 5
 6 005005  104201  000001                      MOV     #1,R1           ; SET UP LOG2 SHIFTER
 7 005007  110201                       1$:     ROL     R1              ; DOUBLE THE TIMEOUT VALUE
 8 005010  103201  000001                      BIC     #1,R1           ; CLEAR THE LOW BIT
 9 005012  117407                              DEC     R0              ; DECREMENT COUNT
10 005013  055007                              BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
11 005014  114007                              CLR     R0              ; CLEAR 9SEC COUNT
12 005015  115407                       2$:    INC     R0              ; INCREMENT 9 SEC COUNT
13 005016  107201  000011                      SUB     #9.,R1          ; SUBTRACT 9 SEC FROM TIMEOUT
14 005020  035015                              BPL     2$              ; IF MORE TIME TO GO, BRANCH
15 005021  000000                              RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                ;RERROR
                                ;
                                ;REPORT ERROR TO HOST PROGRAM
                                ;THIS ROUTINE IS CALLED BY THE ERROR MACROS:
                                ;ERRSF, ERRDF, ERRHRD AND ERRSFT
                                ;
005022                          RERROR: PUSH R0                         ;SAVE ONE REGISTER
005022  100467                                                                  MOV R0,-(SP)
005023  104067                          MOV SP,R0                       ;GET STACK POINTER
005024                                  PUSH <R1,R2,R3,R4>              ;SAVE MORE REGISTERS
005024  100461                                                                  MOV R1,-(SP)
005025  100462                                                                  MOV R2,-(SP)
005026  100463                                                                  MOV R3,-(SP)
005027  100464                                                                  MOV R4,-(SP)
005030  115407                          INC R0                          ;CHANGE SAVED STACK POINTER TO POINT TO
                                                                        ; ADDRESS OF LOCATION AFTER CALL
005031  104271                          MOV (R0)+,R1                    ;GET RETURN PC
005032  104202  001502                  MOV #OUT.01,R2                  ;GET ADDRESS OF WHERE TO PUT DATA
005034  117401                          DEC R1                          ;REDUCE TO PC OF ERROR CALL
005035  100221                          MOV R1,(R2)+                    ;PUT PC IN OUT BUFFER
005036  115401                          INC R1                          ;GET BACK TO RETURN PC
005037  104213                          MOV (R1)+,R3                    ;GET ERROR NUMBER AND TYPE
005040  100223                          MOV R3,(R2)+                    ;PUT IN BUFFER
005041  104303  001152                  MOV LUNIT,R3                    ;PUT UNIT NUMBER IN BUFFER
005043  100223                          MOV R3,(R2)+
005044  104113                          MOV (R1),R3                     ;GET MESSAGE POINTER
005045  103203  170000                  BIC #^C007777,R3                ;CLEAR OTHER BITS
005047  100223                          MOV R3,(R2)+                    ;PUT IN OUT BUFFER
005050  104214                          MOV (R1)+,R4                    ;GET COUNT OF PARAMETERS
                                                                        ;R1 IS NOW POINTING TO INSTRUCTION AFTER ERROR CALL
005051  110704                          SWAB R4                         ;EXTRACT NUMBER OF PARAMETERS FROM ERROR MACRO
005052  110604                          ROR R4
005053  110604                          ROR R4
005054  110604                          ROR R4
005055  110604                          ROR R4
005056  103204  177760                  BIC #177760,R4
005060  104040  005101                  MOV R4,SPADJU+1                 ;SAVE FOR LATER ADJUSTMENT OF STACK POINTER
005062  015067                          BEQ RERRCA                      ;BRANCH IF NO PARAMETERS
005063  104273                  RERRPA: MOV (R0)+,R3                    ;GET PARAMETER
005064  100223                          MOV R3,(R2)+                    ;STORE PARAMETER IN OUT BUFFER
005065  117404                          DEC R4                          ;COUNT THE PARAMETERS
005066  055063                          BNE RERRPA                      ;GET NEXT IF MORE
005067  100471                  RERRCA: MOV R1,-(R0)                    ;PUT RETURN ADDRESS ON STACK
005070  104207  060013                  MOV #ERRMES,R0                  ;SEND ERROR PACKET TO HOST PROGRAM
005072  024652                          CALL HOSTRQ
005073                                  POP <R4,R3,R2,R1,R0>            ;RESTORE REGISTERS
005073  104264                                                                  MOV (SP)+,R4
005074  104263                                                                  MOV (SP)+,R3
005075  104262                                                                  MOV (SP)+,R2
005076  104261                                                                  MOV (SP)+,R1
005077  104267                                                                  MOV (SP)+,R0
005100  105206  000000          SPADJU: ADD #0,SP                       ;ADJUST STACK OVER PARAMETERS
                                                                        ;VALUE CHANGED ABOVE
005102  000000                          RETURN
```

```
      1
```

```
                                        ;MESSAGE STORAGE OVERLAY
     2
     3 005103                           FREE:                                   ; FREE AREA STARTS HERE
     4 005103                                   DMOVLY MS,0
       005103  050473                           .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
     5                                          .NLIST BEX
     6 000000     042     110     117  MS2000: .ASCII\'HOST SPECIFIED UNIT #'D16'' THAT CAN'T BE FOUND.'N\
     7 000031     042     124     105          .ASCII\'TEST2 RESTARING'N\
     8 000042     000                          .BYTE   0
     9 000043     042     103     101  MS2001: .ASCII\'CANNOT RECEIVE VALID DRIVE STATE FROM DRIVE AFTER DRIVE WAS INITED'N\
    10 000105     042     103     110          .ASCII\'CHECK IF DRIVE IS POWERED ON.'N\
    11 000125     000                          .BYTE 0
    12 000126     042     104     122  MS2002: .ASCII\'DRIVE STATE RECEIVED HAS BAD PARITY AFTER DRIVE WAS INITED'N\
    13 000164     000                          .BYTE 0
    14 000165     042     104     122  MS2003: .ASCII\'DRIVE IS NOT ASSERTING RECEIVER READY IN DRIVE STATE AFTER DRIVE WAS INITED
    15 000234     000                          .BYTE 0
    16 000235     042     124     111  MS2004: .ASCII\'TIME-OUT ON SEND OF ECHO COMMAND TO DRIVE'N\
    17 000263     042     040     040          .ASCII\'  ECHO DATA  'H8N\
    18 000274     000                          .BYTE 0
    19 000275     042     105     122  MS2005: .ASCII\'ERROR DURING RECEIVE OF ECHO RESPONSE FROM DRIVE'N\
    20 000326     042     040     040          .ASCII\'  ECHO DATA  'H8N\
    21 000337     000                          .BYTE 0
    22 000340     042     105     103  MS2006: .ASCII\'ECHO COMMAND RESPONDED WITH DIFFERENT DATA'N\
    23 000366     042     040     040          .ASCII\'  ECHO DATA SENT'S6H16N\
    24 000402     042     040     040          .ASCII\'  ECHO DATA RECEIVED  'H16N\
    25 000420     000                          .BYTE 0
    26 000421     042     105     122  MS2007: .ASCII\'ERROR BIT SET IN GET STATUS RESPONSE AFTER DRIVE CLEAR COMMAND'N\
    27 000461     042     040     040          .ASCII\'  GET STATUS RESPONSE  'NR1\
    28 000477     000                          .BYTE 0
    29 000500     042     124     111  MS2008: .ASCII\'TIME-OUT ON SEND OF ONLINE COMMAND TO DRIVE'N\
    30 000527     000                          .BYTE 0
    31 000530     042     105     122  MS2009: .ASCII\'ERROR DURING RECEIVE OF ONLINE RESPONSE FROM DRIVE'R1\
    32 000563     000                          .BYTE 0
    33 000564     042     117     116  MS2010: .ASCII\'ONLINE COMMAND WAS UNSUCCESSFUL'NR1\
    34 000606     000                          .BYTE 0
    35 000607     042     117     116  MS2011: .ASCII\'ONLINE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
    36 000642     042     040     040          .ASCII\'  EXPECTED RESPONSE  'H8N\
    37 000657     042     040     040          .ASCII\'  ACTUAL RESPONSE'S4H8N\
    38 000673     000                          .BYTE 0
    39 000674     042     124     111  MS2012: .ASCII\'TIME-OUT ON SEND OF GET UNIT CHARACTERISTICS COMMAND TO DRIVE'N\
    40 000734     000                          .BYTE   0
    41 000735     042     105     122  MS2013: .ASCII\'ERROR DURING RECEIVE OF GET UNIT CHARACTERISTICS COMMAND FROM DRIVE'R1\
    42 001000     000                          .BYTE   0
    43 001001     042     107     105  MS2014: .ASCII\'GET UNIT CHARACTERISTICS COMMAND WAS UNSUCCESSFUL'NR1\
    44 001034     000                          .BYTE   0
    45 001035     042     107     105  MS2015: .ASCII\'GET UNIT CHARACTERISTICS COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
    46 001101     042     040     040          .ASCII\'  EXPECTED RESPONSE  'H8N\
    47 001116     042     040     040          .ASCII\'  ACTUAL RESPONSE'S4H8N\
    48 001132     000                          .BYTE   0
    49 001133     042     040     040  MS2016: .ASCII\'  HOST PROGRAM GAVE DM CODE IMPROPER DATA'N\
    50 001161     042     040     040          .ASCII\'  EXPECTED VALUE SHOULD BE BETWEEN 0 AND 3'N\
    51 001207     042     040     040          .ASCII\'  ACTUAL VALUE WAS 'D8N\
    52 001223     000                          .BYTE 0
    53 001224     042     124     111  MS2017: .ASCII\'TIME-OUT ON SEND OF DIAGNOSE COMMAND TO DRIVE'N\
    54 001254     000                          .BYTE 0
    55 001255     042     105     122  MS2018: .ASCII\'ERROR DURING RECEIVE OF DIAGNOSE RESPONSE FROM DRIVE'R1\
    56 001311     000                          .BYTE 0
```

```
     57 001312      042      104      111  MS2019: .ASCII\'DIAGNOSE COMMAND WAS UNSUCCESSFUL'NR1\
     58 001335      000                            .BYTE   0
     59 001336      042      104      111  MS2020: .ASCII\'DIAGNOSE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
     60 001372      042      040      040          .ASCII\''  EXPECTED RESPONSE  ''H8N\
     61 001407      042      040      040          .ASCII\''  ACTUAL RESPONSE''S4H8N\
     62 001423      000                            .BYTE 0
     63 001424      042      104      122  MS2021: .ASCII\'DRIVE DIAGNOSTIC REPORTS A HARD ERROR'N\
     64 001450      042      040      040          .ASCII\''  TEST NUMBER  ''H16N\
     65 001462      042      040      040          .ASCII\''  DRIVE TYPE   ''H8N\
     66 001474      042      040      040          .ASCII\''  ERROR NUMBER ''H16N\
     67 001507      122      062                   .ASCII\R2\
     68 001510      000                            .BYTE 0
     69 001511      042      110      117  MS2022: .ASCII\'HOST PROGRAM DOWN LINE LOADED A DIAGNOSTIC WITH A ZERO BYTE COUNT'N\
     70 001553      000                            .BYTE 0
     71 001554      042      104      111  MS2023: .ASCII\'DIAGNOSTIC ''A6'' REQUESTED BY THE DRIVE COULD NOT BE SUPPLIED BY HOST.'N\
     72 001620      000                            .BYTE   0
     73 001621      042      124      111  MS2024: .ASCII\'TIME-OUT ON SEND OF MEMORY READ COMMAND TO DRIVE'N\
     74 001652      000                            .BYTE 0
     75 001653      042      105      122  MS2025: .ASCII\'ERROR DURING RECEIVE OF MEMORY READ RESPONSE FROM DRIVE'NR1\
     76 001711      000                            .BYTE 0
     77 001712      042      115      105  MS2026: .ASCII\'MEMORY READ COMMAND WAS UNSUCCESSFUL'NR1\
     78 001736      000                            .BYTE   0
     79 001737      042      115      105  MS2027: .ASCII\'MEMORY READ COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
     80 001775      042      040      040          .ASCII\''  EXPECTED RESPONSE  ''H8N\
     81 002012      042      040      040          .ASCII\''  ACTUAL RESPONSE''S4H8N\
     82 002026      000                            .BYTE 0
     83 002027      042      124      111  MS2028: .ASCII\'TIME-OUT ON SEND OF MEMORY WRITE COMMAND TO DRIVE'N\
     84 002061      000                            .BYTE 0
     85 002062      042      105      122  MS2029: .ASCII\'ERROR DURING RECEIVE OF MEMORY WRITE RESPONSE FROM DRIVE'R1\
     86 002120      000                            .BYTE 0
     87 002121      042      115      105  MS2030: .ASCII\'MEMORY WRITE COMMAND WAS UNSUCCESSFUL'NR1\
     88 002146      000                            .BYTE   0
     89 002147      042      115      105  MS2031: .ASCII\'MEMORY WRITE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
     90 002205      042      040      040          .ASCII\''  EXPECTED RESPONSE  ''H8N\
     91 002222      042      040      040          .ASCII\''  ACTUAL RESPONSE''S4H8N\
     92 002236      000                            .BYTE   0
     93 002237      042      124      111  MS2032: .ASCII\'TIME-OUT ON SEND OF RUN COMMAND TO DRIVE'N\
     94 002264      000                            .BYTE 0
     95 002265      042      105      122  MS2033: .ASCII\'ERROR DURING RECEIVE OF RUN RESPONSE FROM DRIVE'R1\
     96 002316      000                            .BYTE 0
     97 002317      042      122      125  MS2034: .ASCII\'RUN COMMAND WAS UNSUCCESSFUL'NR1\
     98 002337      000                            .BYTE 0
     99 002340      042      122      125  MS2035: .ASCII\'RUN COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
    100 002372      042      040      040          .ASCII\''  EXPECTED RESPONSE  ''H8N\
    101 002407      042      040      040          .ASCII\''  ACTUAL RESPONSE''S4H8N\
    102 002423      000                            .BYTE 0
    103 002424      042      124      111  MS2036: .ASCII\'TIME-OUT ON SEND OF RECALIBRATE COMMAND TO DRIVE'N\
    104 002455      000                            .BYTE 0
    105 002456      042      105      122  MS2037: .ASCII\'ERROR DURING RECEIVE OF RECALIBRATE RESPONSE FROM DRIVE'R1\
    106 002513      000                            .BYTE 0
    107 002514      042      122      105  MS2038: .ASCII\'RECALIBRATE COMMAND WAS UNSUCCESSFUL'NR1\
    108 002540      000                            .BYTE 0
    109 002541      042      122      105  MS2039: .ASCII\'RECALIBRATE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
    110 002577      042      040      040          .ASCII\''  EXPECTED RESPONSE  ''H8N\
    111 002614      042      040      040          .ASCII\''  ACTUAL RESPONSE''S4H8N\
    112 002630      000                            .BYTE 0
    113 002631      042      124      111  MS2040: .ASCII\'TIME-OUT ON SEND OF GET STATUS COMMAND TO DRIVE'N\
```

```
002662   000                      BYTE 0
002663   042   105   122  MS2041   ASCII\''ERROR DURING RECEIVE OF GET STATUS RESPONSE FROM DRIVE''R1\
002720   000                      .BYTE 0
002721   042   107   105  MS2042: .ASCII\''GET STATUS COMMAND WAS UNSUCCESSFUL''NR1\
002745   000                      .BYTE 0
002746   042   107   105  MS2043: .ASCII\''GET STATUS COMMAND DID NOT RETURN EXPECTED RESPONSE CODE''N\
003003   042   040   040          .ASCII\''  EXPECTED RESPONSE  ''H8N\
003020   042   040   040          .ASCII\''  ACTUAL RESPONSE''S4H8N\
003034   000                      .BYTE 0
003035   042   124   111  MS2044: .ASCII\''TIME-OUT ON SEND OF DRIVE CLEAR COMMAND TO DRIVE''N\
003066   000                      .BYTE 0
003067   042   105   122  MS2045: .ASCII\''ERROR DURING RECEIVE OF DRIVE CLEAR RESPONSE FROM DRIVE''R1\
003124   000                      .BYTE 0
003125   042   104   122  MS2046: .ASCII\''DRIVE CLEAR COMMAND WAS UNSUCCESSFUL''NR1\
003151   000                      .BYTE 0
003152   042   104   122  MS2047: .ASCII\''DRIVE CLEAR COMMAND DID NOT RETURN EXPECTED RESPONSE CODE''N\
003210   042   040   040          .ASCII\''  EXPECTED RESPONSE  ''H8N\
003225   042   040   040          .ASCII\''  ACTUAL RESPONSE''S4H8N\
003241   000                      .BYTE 0
003242   101   064   116  MSG1:   .ASCII\A4N\
003243   042   111   116          .ASCII\''INFORMATION SENT BACK FROM THE DRIVE IS BEING PRESENTED.''N\
003301   042   040   040          .ASCII\''  TEST NUMBER  ''H16N\
003313   042   040   040          .ASCII\''  DRIVE TYPE   ''H8N\
003325   042   040   040          .ASCII\''  ERROR NUMBER ''H16N\
003340   122   062                .ASCII\R2\
003341   000                      .BYTE 0
003342   042   106   117  MSG2:   .ASCII\''FOLLOWING REPORT HAS BEEN TRUNCATED DUE TO SIZE''N\
003373   000                      .BYTE 0
003374   116   042   124  MSR0:   .ASCII\N''TIMEOUT ERROR OCCURED DURING RECEIVE XFC''N\
003422   000                      .BYTE   0
003423   116   042   061  MSR1:   .ASCII\N''1ST WORD NOT START FRAME DURING RECEIVE XFC''N\
003452   000                      .BYTE   0
003453   116                MSR2:   .ASCII\N\
003453   042   106   122          .ASCII\''FRAMING ERROR OCCURED ON SDI LEVEL 0 READ DURING RECEIVE XFC''N\
003513   000                      .BYTE   0
003514   116                MSR3:   .ASCII\N\
003514   042   103   110          .ASCII\''CHECKSUM ERROR OCCURED ON SDI LEVEL 0 READ DURING RECEIVE XFC''N\
003554   000                      .BYTE   0
003555   116                MSR4:   .ASCII\N\
003555   042   102   125          .ASCII\''BUFFER SIZE SMALLER THEN RESPONSE DURING RECEIVE XFC''N\
003611   000                      .BYTE   0
003612   116                MSR5:   .ASCII\N\
003612   042   103   117          .ASCII\''CODE FROM RECEIVE XFC WAS UNINTELLIGIBLE FROM SUBSYSTEM  ''H16N\
003652   000                      .BYTE   0
003653   042   125   116  ER5000: .ASCII  \''UNABLE TO FIND REQUESTED DRIVE FOR TESTING''N\
003701   042   124   110          .ASCII  \''THE FOLLOWING IS VISIBLE ON THE PORTS''N\
003725   042   125   104          .ASCII  \''UDA PORT 0 -- ''R1\
003736   042   125   104          .ASCII  \''UDA PORT 1 -- ''R1\
003747   042   125   104          .ASCII  \''UDA PORT 2 -- ''R1\
003760   042   125   104          .ASCII  \''UDA PORT 3 -- ''R1\
003771   000                      .BYTE   0
003772   042   116   117  SER10:  .ASCII  \''NO DRIVE ATTACHED''N\
004004   000                      .BYTE   0
004005   042   122   103  SER11:  .ASCII  \''RCVR RDY NEVER ASSERTED''N\
004022   000                      .BYTE   0
004023   042   124   111  SER12:  .ASCII  \''TIMEOUT OF SEND''N\
004034   000                      .BYTE   0
```

```
171 004035      042     124     111  SER13:  .ASCII  \"TIMEOUT OF RECEIVE'N\
172 004047      000                          .BYTE   0
173 004050      042     106     111  SER14:  .ASCII  \"FIRST WORD RECEIVED WAS NOT START FRAME'N\
174 004075      000                          .BYTE   0
175 004076      042     106     122  SER15:  .ASCII  \"FRAMING ERROR ON LEVEL 0 RECEIVE'N\
176 004117      000                          .BYTE   0
177 004120      042     103     110  SER16:  .ASCII  \"CHECKSUM ERROR ON LEVEL 0 RECEIVE'N\
178 004142      000                          .BYTE   0
179 004143      042     122     105  SER17:  .ASCII  \"RESPONSE LONGER THAN EXPECTED FOR CMD'N\
180 004167      000                          .BYTE   0
181 004170      042     104     122  SER18:  .ASCII  \"DRIVE "R1\
182 004175      000                          .BYTE   0
183 004176      104     061     062  SER18D: .ASCII  \D12",  "\       ; NOTE: THE STRING WITHIN THE \\'S <<MUST>>
184 004202      104     061     062  SER18C: .ASCII  \D12",  "\       ; BE AN EVEN NUMBER OF CHAR, BECAUSE THE
185 004206      104     061     062  SER18B: .ASCII  \D12",  "\       ; LABELS WILL FORCE WORD ALINGMENT, LEAVING
186 004212      104     061     062  SER18A: .ASCII  \D12N\           ; JUNK IN THE LAST BYTE. (SER18A OK TO BE ODD)
187 004214      000                          .BYTE   0
188 004215      042     104     122  SER40:  .ASCII  \"DRIVE NOT AVAILABLE TO THIS UDA'N\
189 004236      000                          .BYTE   0
190 004237      042     125     116  SER41:  .ASCII  \"UNSPINABLE DRIVE "R1\
191 004251      000                          .BYTE   0
192 004252      042     122     105  STAT0:  .ASCII\"REAL TIME STATE"S2H16N\
193 004265      042     123     124          .ASCII\"STATUS (R TO L)"S2H16S2H16S2H16S2H16S2H16S2H16S2H16N\
194 004320      000                          .BYTE   0
195 004320                                   DMEND
    004321  013364                           .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
196         000001                           .END
```

SYMBOL TABLE

```
ARGS$ = 000003   DRVID = 000004   GETSTA= 000011   IN.25   001575   MS2021  001424
ATTN  = 000002   DRVONL= 000213   GETSUB= 000210   IN.26   001576   MS2022  001511
AVAIL = 000100   DRVRUN= 000014   GRPCYL= 000002   IN.27   001577   MS2023  001554
BF.DAT= 000000   DU.DFL= 020000   GRPOFF= 000011   IN.28   001600   MS2024  001621
BF.ECC= 000401   DU.FTL= 050000   GSTS3   002505   IN.29   001601   MS2025  001653
BF.EDC= 000400   DU.INF= 030000   GSTS6   002506   IN.30   001602   MS2026  001712
BREAK = 000000   DU.QUE= 010000   GSTS7   002575   IN.31   001603   MS2027  001737
BUFFLG= 040000   DU.SPC= 060000   GTSTAT  004140   IN.32   001604   MS2028  002027
BUFSIZ= 000043   DU.TER= 040000   GT.CMD  002726   IN.33   001605   MS2029  002062
BUF1    001222   D.LIMT= 000001   HBHINB= 007777   IN.34   001606   MS2030  002121
CHECK = 000010   D.SCHR= 000002   HBLONB= 170377   IRECLB= 000216   MS2031  002147
CHGMOD= 000201   ECC   = 000015   HDRPRE= 000005   LARGE = 000001   MS2032  002237
CHRRES= 000170   ECCFLG= 010000   HD.BAD= 110000   LBHINB= 177417   MS2033  002265
CLRBUF  004703   ECCRSH= 000002   HD.DBN= 140000   LBLONB= 177760   MS2034  002317
CLRDRV  004220   ECHO  = 000010   HD.LBN= 000000   LBNCYL= 000000   MS2035  002340
COMPAR= 000006   ECHOC = 000350   HD.PRV= 050000   LBNHST= 000012   MS2036  002424
COMPLT= 000176   ECHOD   001163   HD.RBN= 060000   LBNTRK= 000011   MS2037  002456
CONMEM  004325   ECHO1   002431   HD.REV= 030000   LINKLN= 000007   MS2038  002514
CR.DIA  001455   ECHO1A  002433   HD.XBN= 120000   LOBYTE= 000377   MS2039  002541
CR.DRC  001177   ECHO2   002434   HEADER= 000002   LONG    001455   MS2040  002631
CR.GCR  001440   ECHO?   002453   HIBYTE= 177400   LONGTO= 000001   MS2041  002663
CR.GST  001171   ECHC    002460   HICYL = 000001   LOW   = 000000   MS2042  002721
CR.INR  001473   ECHO5   002474   HIDBN = 000003   LUNIT   001152   MS2043  002746
CR.ONL  001446   EOC   = 100000   HILBN = 000002   MAXSND= 001750   MS2044  003035
CR.RDM  001422   ERECOV= 000006   HIMEM = 007774   MEDTYP= 000006   MS2045  003067
CR.RUN  001465   ERHARD= 100000   HIRBN = 000003   MESSAG= 060015   MS2046  003125
CR.WRM  001206   ERLEV = 000002   HIXBN = 000002   MICREV= 000003   MS2047  003152
CVT   = 000020   ERRMC = 060014   HOSTRQ  004652   MRD   = 000016   MWR   = 000017
DATAVL= 040000   ERRMES= 060013   HRDREV= 000003   MSG1    003242   NAM.1   001145
DATPRE= 000005   ERRN  = 004000   ILOOP   002400   MSG2    003342   NAM.2   001146
DBNCYL= 000022   ERRTYP= 100000   INR     001500   MSR0    003374   NAM.3   001147
DIA     001462   ERSOFT= 140000   INSEEK= 000012   MSR1    003423   NUMPTR= 000003
DIAGDR  003102   ER.END  004213   IN.RQ   001544   MSR2    003453   OK.END  004216
DIAGNO  003075   ER5000  003653   IN.01   001545   MSR3    003514   ONL     001453
DIAGNS  003055   EXIT  = 000021   IN.02   001546   MSR4    003555   OUT.RQ  001501
DIAG.1  001144   EX.END  004217   IN.03   001547   MSR5    003612   OUT.01  001502
DIA.EN= 000374   FB.DAT= 000000   IN.04   001550   MS2000  000000   OUT.02  001503
DIA.OP= 000003   FB.EDC= 000400   IN.05   001551   MS2001  000043   OUT.03  001504
DINIT = 000011   FCTSIZ= 000010   IN.06   001552   MS2002  000126   OUT.04  001505
DISCON= 000204   FFFC  = 177774   IN.07   001553   MS2003  000165   OUT.05  001506
DIV50   004553   FFFD  = 177775   IN.08   001554   MS2004  000235   OUT.06  001507
DONE  = 060016   FFFE  = 177776   IN.09   001555   MS2005  000275   OUT.07  001510
DONECD  002367   FNDASC  004567   IN.10   001556   MS2006  000340   OUT.08  001511
DO.DCL  003270   FORMAT= 000001   IN.11   001557   MS2007  000421   OUT.09  001512
DO.DC4  003265   FRAME = 000004   IN.12   001560   MS2008  000500   OUT.10  001513
DO.DIX  003315   FREE    005103   IN.13   001561   MS2009  000530   OUT.11  001514
DO.DI0  003612   FSTOP = 100000   IN.14   001562   MS2010  000564   OUT.12  001515
DO.DI1  003347   FTLDEV= 040000   IN.15   001563   MS2011  000607   OUT.13  001516
DO.DI2  003377   FTLSYS= 000000   IN.16   001564   MS2012  000674   OUT.14  001517
DO.DI3  003530   FT.BUF= 000000   IN.17   001565   MS2013  000735   OUT.15  001520
DO.DI7  003534   FT.HI = 000001   IN.18   001566   MS2014  001001   OUT.16  001521
DO.DI8  003537   FT.LOW= 000002   IN.19   001567   MS2015  001035   OUT.17  001522
DRC     001204   GCC.EN= 000170   IN.20   001570   MS2016  001133   OUT.18  001523
DRCBYT  001205   GCR     001445   IN.21   001571   MS2017  001224   OUT.19  001524
DRCLR1  002504   GETBCN  004365   IN.22   001572   MS2018  001255   OUT.20  001525
DRTYPE= 000007   GETCHR= 000207   IN.23   001573   MS2019  001312   OUT.21  001526
DRVCLR= 000005   GETST   001176   IN.24   001574   MS2020  001336   OUT.22  001527
```

SYMBOL TABLE

```
OUT.23  001530          RDM     001427          SEND  = 000004          ST.DIA  002665          T2STRT  002231
OUT.24  001531          RDMEM   003642          SENDHR  001151          ST.DR = 000040          T4BB1 = 060005
OUT.25  001532          RDM.EN= 000162          SER10   003772          ST.ERR= 000002          T4BB2 = 060006
OUT.26  001533          RDM.OP= 000215          SER11   004005          ST.FE = 000200          T4MPRM= 060003
OUT.27  001534          RDSTAT  004621          SER12   004023          ST.FO = 002000          T4MXFR= 060011
OUT.28  001535          READ    003021          SER13   004035          ST.MOD= 000001          T4SEEK= 060010
OUT.29  001536          RECAL   004064          SER14   004050          ST.MSK= 000000          T4SOFT= 060007
OUT.30  001537          REGS$ = 177777          SER15   004076          ST.OA = 000200          T4UPRM= 060004
OUT.31  001540          REGU$ = 000001          SER16   004120          ST.PE = 000040          UDADM2= 001000 G
OUT.32  001541          RERRCA  005067          SER17   004143          ST.PS = 000002          UDA50 = 000001
OUT.33  001542          RERROR  005022          SER18   004170          ST.RE = 000100          UDA52 = 000000
OUT.34  001543          RERRPA  005063          SER18A  004212          ST.RR = 000100          UNITNB  000740
OVERFL= 000002          RETS  = 000001          SER18B  004206          ST.RTY= 000003          UNITS   000720
OVE.MN= 000714          REVECT= 000004          SER18C  004202          ST.RU = 000001          UNIT0 = 000001
OVE.MS= 000000          REVS  = 000007          SER18D  004176          ST.SR = 000020          UNIT1 = 000002
OVL.MN= 004170          RM    = 000004          SER18E  001157          ST.STA= 000001          UNIT2 = 000004
OVL.MS= 004322          RREAL = 013400          SER40   004215          ST.S7 = 000400          UNIT3 = 000010
OVL...= 010512          RSTOP = 100000          SER41   004237          ST.UNT= 000000          UNSSUC= 000175
OVSTRT= 007774          RUN     001472          SHRTTO= 000000          ST.WE = 000010          UREAD = 000013
OVS.MN= 001040          RUNDR   004010          SNDAGN  004657          SUB   = 000013          UTOTST= 060012
OVS.MS= 011420          RWRDY = 100000          SPADJU  005100          TALKER  004712          UWRITE= 000014
OV...  = 011132         RW.ANG= 000006          SS    = 000001          TALK1A  004724          WAITSI= 000012
PHYSA = 001000          RW.BUF= 000001          ST      000744          TALK1B  004734          WBUFLN= 000401
PORT0   002315          RW.CMD= 000004          STACK   001647          TALK2A  004771          WCONT = 040000
PORT2   002316          RW.HI = 000003          START   001650          TALK2B  004772          WREAL = 122400
PORT3   002346          RW.LOW= 000002          STATEX  004647          TEST    002373          WRITE   002755
PORT4   002354          RW.SDI= 000005          STATLP  004625          TESTEX  003101          WRM     001213
PORT5   002363          RW.STA= 000000          STATOK  004640          TESTX   002357          WRM.OP= 000017
PRMS  = 000002          SAFWRD  001156          STATUS= 000007          TIMEOU= 000001          WRONG = 000002
PTYPE$= 000020          SAVREG  001153          STAT0   004252          T0      005005          WRTMEM  003730
RBNTRK= 000004          SAVRID  000743          STOSTA  004274          TOOBIG= 000001          WSTOP = 140000
RBUFLN= 000415          SAVSTA  000742          STSIZE= 000200          TRKGRP= 000003          XBNCYL= 000021
RCONT = 000000          SBCRES= 000167          STSRES= 000366          TYPERR  004513          XFERRT= 000000
RCTCPS= 000001          SDI     000741          ST.C  = 000002          T00     002647          XMTERR= 000400
RCTCSZ= 000014          SDILTO  001155          ST.CON= 000002          T1MSIZ= 060000          XOR     004775
RCV   = 000005          SDISTO  001154          ST.DB = 001000          T2CMD = 060002          XREAD = 000002
RCVERR= 000004          SDIVER= 000000          ST.DF = 000020          T2DLL = 060001          XWRITE= 000003
RCVRDY= 000001

. ABS.  022264      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  3855 WORDS  ( 16 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  70 PAGES
B:UDAT2.050,B:UDAT2.L50/C=$DMACRO,B:UDAT2
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

UDAT2 DISK RESIDENT DMACR X04.01 19-AUG-82 13:10:12 PAGE S-5 SEQ 0317
CROSS REFERENCE TABLE (CREF V04.00)

MS2001	29-26	61-9#
MS2002	29-28	61-12#
MS2003	29-37	61-14#
MS2004	30-24	61-16#
MS2005	30-26	61-19#
MS2006	30-33	61-22#
MS2007	31-27	61-26#
MS2008	31-38	61-29#
MS2009	31-41	61-31#
MS2010	31-48	61-33#
MS2011	31-50	61-35#
MS2012	32-10	61-39#
MS2013	32-13	61-41#
MS2014	32-20	61-43#
MS2015	32-22	61-45#
MS2016	33-36	61-49#
MS2017	37-23	61-53#
MS2018	37-27	61-55#
MS2019	37-38	61-57#
MS2020	37-40	61-59#
MS2021	38-29	61-63#
MS2022	40-30	61-69#
MS2023	41-25	61-71#
MS2024	43-30	61-73#
MS2025	43-34	61-75#
MS2026	43-42	61-77#
MS2027	43-44	61-79#
MS2028	44-26	61-83#
MS2029	44-30	61-85#
MS2030	44-38	61-87#
MS2031	44-40	61-89#
MS2032	45-9	61-93#
MS2033	45-12	61-95#
MS2034	45-19	61-97#
MS2035	45-21	61-99#
MS2036	46-9	61-103#
MS2037	46-12	61-105#
MS2038	46-19	61-107#
MS2039	46-21	61-109#
MS2040	47-18	61-113#
MS2041	47-21	61-115#
MS2042	47-29	61-117#
MS2043	47-31	61-119#
MS2044	48-18	61-123#
MS2045	48-21	61-125#
MS2046	48-29	61-127#
MS2047	48-31	61-129#
MSG1	38-33	61-133#
MSG2	51-53	61-140#
MSR0	52-22	61-142#
MSR1	52-26	61-144#
MSR2	52-30	61-146#
MSR3	52-34	61-149#
MSR4	52-38	61-152#
MSR5	52-40	61-155#
MWR	3-18#	55-21

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

PORT$
28-18 28-20 28-22 28-40#

PRM$
27-41 27-41 27-41 27-41 27-41 27-41 27-41 27-41 27-41 27-41 27-41# 29-26 29-26 29-26
29-26 29-26 29-26 29-26 29-26 29-26 29-26 29-26# 29-28 29-28 29-28 29-28 29-28 29-28
29-28 29-28 29-28 29-28 29-28# 29-37 29-37 29-37 29-37 29-37 29-37 29-37 29-37 29-37
29-37 29-37# 30-24 30-24 30-24 30-24 30-24 30-24 30-24 30-24 30-24 30-24 30-24# 30-26
30-26 30-26 30-26 30-26 30-26 30-26 30-26 30-26 30-26 30-26# 30-33 30-33 30-33 30-33
30-33 30-33 30-33 30-33 30-33 30-33 30-33# 31-27 31-27 31-27 31-27 31-27 31-27 31-27
31-27 31-27 31-27 31-27# 31-38 31-38 31-38 31-38 31-38 31-38 31-38 31-38 31-38 31-38
31-38# 31-41 31-41 31-41 31-41 31-41 31-41 31-41 31-41 31-41 31-41 31-41# 31-48 31-48
31-48 31-48 31-48 31-48 31-48 31-48 31-48 31-48 31-48# 31-50 31-50 31-50 31-50 31-50
31-50 31-50 31-50 31-50 31-50 31-50# 32-10 32-10 32-10 32-10 32-10 32-10 32-10 32-10
32-10 32-10 32-10# 32-13 32-13 32-13 32-13 32-13 32-13 32-13 32-13 32-13 32-13 32-13#
32-20 32-20 32-20 32-20 32-20 32-20 32-20 32-20 32-20 32-20 32-20# 32-22 32-22 32-22
32-22 32-22 32-22 32-22 32-22 32-22 32-22 32-22# 33-36 33-36 33-36 33-36 33-36 33-36
33-36 33-36 33-36# 37-23 37-23 37-23 37-23 37-23 37-23 37-23 37-23 37-23 37-23 37-27# 37-38
37-23 37-23# 37-27 37-27 37-27 37-27 37-27 37-27 37-27 37-27 37-27 37-27 37-27 37-38
37-38 37-38 37-38 37-38 37-38 37-38 37-38 37-38 37-38 37-38# 37-40 37-40 37-40 37-40
37-40 37-40 37-40 37-40 37-40 37-40 37-40# 38-29 38-29 38-29 38-29 38-29 38-29 38-29
38-29 38-29 38-29 38-29# 40-30 40-30 40-30 40-30 40-30 40-30 40-30 40-30 40-30 40-30
40-30# 41-25 41-25 41-25 41-25 41-25 41-25 41-25 41-25 41-25 41-25 41-25# 43-30 43-30
43-30 43-30 43-30 43-30 43-30 43-30 43-30 43-30# 43-34 43-34 43-34 43-34 43-34 43-34
43-34 43-34 43-34 43-34# 43-42 43-42 43-42 43-42 43-42 43-42 43-42 43-42 43-42 43-42
43-42 43-42 43-42# 43-44 43-44 43-44 43-44 43-44 43-44 43-44 43-44 43-44 43-44 43-44#
44-26 44-26 44-26 44-26 44-26 44-26 44-26 44-26 44-26 44-26 44-26# 44-30 44-30 44-30
44-30 44-30 44-30 44-30 44-30 44-30 44-30 44-30# 44-38 44-38 44-38 44-38 44-38 44-38
44-38 44-38 44-38 44-38# 44-40 44-40 44-40 44-40 44-40 44-40 44-40 44-40 44-40 44-40
44-40 44-40# 45-9 45-9 45-9 45-9 45-9 45-9 45-9 45-9 45-9 45-9 45-9# 45-12
45-12 45-12 45-12 45-12 45-12 45-12 45-12 45-12 45-12 45-12 45-12# 45-19 45-19 45-19 45-19
45-19 45-19 45-19 45-19 45-19 45-19 45-19# 45-21 45-21 45-21 45-21 45-21 45-21 45-21
45-21 45-21 45-21 45-21# 46-9 46-9 46-9 46-9 46-9 46-9 46-9 46-9 46-9 46-9
46-9# 46-12 46-12 46-12 46-12 46-12 46-12 46-12 46-12 46-12 46-12 46-12# 46-19 46-19
46-19 46-19 46-19 46-19 46-19 46-19 46-19 46-19# 46-21 46-21 46-21 46-21 46-21
46-21 46-21 46-21 46-21 46-21 46-21# 47-18 47-18 47-18 47-18 47-18 47-18 47-18 47-18
47-18 47-18 47-18# 47-21 47-21 47-21 47-21 47-21 47-21 47-21 47-21 47-21 47-21 47-21#
47-29 47-29 47-29 47-29 47-29 47-29 47-29 47-29 47-29 47-29 47-29# 47-31 47-31 47-31
47-31 47-31 47-31 47-31 47-31 47-31 47-31 47-31# 48-18 48-18 48-18 48-18 48-18 48-18
48-18 48-18 48-18# 48-21 48-21 48-21 48-21 48-21 48-21 48-21 48-21 48-21 48-21
48-21 48-21# 48-29 48-29 48-29 48-29 48-29 48-29 48-29 48-29 48-29 48-29# 48-31
48-31 48-31 48-31 48-31 48-31 48-31 48-31 48-31 48-31 48-31#

PTYPE$
27-41 27-41 27-41# 30-24 30-24 30-24# 30-26 30-26 30-26# 30-33 30-33 30-33 30-33 30-33#
30-33# 31-41 31-41 31-41 31-41 31-41# 31-41# 31-50 31-50 31-50 31-50 31-50# 31-50# 32-13
32-13 32-13 32-13 32-13# 32-13# 32-22 32-22 32-22 32-22 32-22# 32-22# 33-36 33-36 33-36#
37-27 37-27 37-27 37-27 37-27# 37-27# 37-40 37-40 37-40 37-40 37-40# 37-40# 41-25 41-25
41-25 41-25 41-25 41-25# 41-25# 41-25# 43-34 43-34 43-34 43-34 43-34# 43-34# 43-44
43-44 43-44 43-44 43-44# 43-44# 44-30 44-30 44-30 44-30 44-30# 44-30# 44-40 44-40 44-40
44-40 44-40# 44-40# 45-12 45-12 45-12 45-12 45-12# 45-12# 45-21 45-21 45-21 45-21 45-21#
45-21# 46-12 46-12 46-12 46-12 46-12# 46-12# 46-21 46-21 46-21 46-21 46-21# 46-21# 47-21
47-21 47-21 47-21 47-21# 47-21# 47-31 47-31 47-31 47-31 47-31# 47-31# 48-21 48-21 48-21
48-21 48-21# 48-21# 48-31 48-31 48-31 48-31 48-31# 48-31#

RBNTRK
4-31#

RBUFLN
2-156#

RCONT
2-128#

RCTCPS
4-8#

RCTCSZ
4-40#

RCV
3-8# 23-57 56-35

RCVERR
5-23# 54-17

CROSS REFERENCE TABLE (CREF V04.00)

```
STATLP  54-9#    54-14    54-20
STATOK  54-12    54-17#
STATUS   3-10#   49-10    54-9
STOSTA  31-26    31-47    32-19    37-37    43-41    44-37    45-18    46-18    47-28    48-28    49-4#
STSIZE  19-20#   20-25    40-32    40-34
STSRES   5-45#   47-24    47-31    47-31
SUB      4-22#
T00     32-11    32-14    32-16    32-21    32-24#
T1MSIZ   5-3#
T2CMD    5-5#    33-24
T2DLL    5-4#    40-16
T2STRT  27-16#
T4BB1    5-8#
T4BB2    5-9#
T4MPRM   5-6#
T4MXFR   5-12#   56-31
T4SEEK   5-11#
T4SOFT   5-10#
T4UPRM   5-7#
TALK1A  56-17    56-20#
TALK1B  56-23    56-25#   56-43
TALK2A  56-37    56-46#
TALK2B  56-19    56-41    56-45    56-47#
TALKER  31-34    32-6     37-16    43-24    44-20    45-5     46-5     47-11    48-11    56-12#
TEST    28-35    29-11#
TESTEX  33-29    36-22#
TESTX   28-37#   36-22
TIMEOU   4-61#
TO      32-26    32-30    58-1#
TOOBIG   4-69#
TRKGRP   4-28#
TYPERR  31-40    32-12    37-26    43-33    44-29    45-11    46-11    47-20    48-20    52-16#
UDA50    2-32#    2-47     2-70     2-91
UDA52    2-33#    2-38     2-55     2-65     2-86    18-10    59-46
UDADM2   2-72#
UNIT0    4-46#   27-26    28-25
UNIT1    4-47#
UNIT2    4-48#
UNIT3    4-49#
UNITNB  19-11#   27-50*   28-23*   28-39    41-3*
UNITS   19-6#    23-14    25-7     25-19    25-26    26-7     26-36    27-24    28-13
UNSSUC   5-42#   31-45    32-17    37-35    43-39    44-35    45-16    46-16    47-26    48-26
UREAD    3-14#   40-35
UTOTST   5-13#   25-4
UWRITE   3-15#
WAITSI   3-13#
WBUFLN   2-155#   2-156
WCONT    2-126#
WREAL    2-130#
WRITE   33-31    34-10#
WRM     20-19    20-20#   33-6*    33-7*    33-8*    33-9*    33-10*   34-11*   34-12*   34-17*   34-18*   34-19*   40-4*    40-21*
        40-40*   40-41*   40-45*   40-50    40-58*   40-75*
WRM.OP  19-48#   20-20
WRONG    4-65#
WRTMEM  33-12    34-22    40-54    44-18#
WSTOP    2-125#
```

CROSS REFERENCE TABLE (CREF V04.00)

TABLE OF CONTENTS

```
                                .TITLE  UDAT3   DISK FUNCTIONAL

                        ;
                        ;
                        ; COPYRIGHT (C)  1981
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
000000                  UDA50=0
000001                  UDA52=1
                        ;THIS PROGRAM SHALL BE ASSEMBLED WITH THE PROGRAM DMACRO
                        ;USING A COMMAND LINE SIMILAR TO:
                        ;
                        ;  UDAT3,UDAT3/C=[1,2]DMACRO,UDAT3
                        ;
                        ;THEN LINK THE OBJECT FILE USING A COMMAND LINE SIMILAR TO:
                        ;
                        ;  UDAT3.BIN/L=UDAT3
                        ;
000000                  TEST4   =       0                       ; THIS IS NOT TEST4
```

```
                                        .SBTTL  START OF TEST CODE
                                        .MCALL  DMCODE,DMEND,DMOVLY
                                        .MCALL  JMP,BR,BEQ,CALL,BPL,BCC,BNE,BMI,RETURN
                                        .MCALL  DMODT

 000000                                 DMCODE  UDADM3,0,1364,3,0,1000

 001364  000000                         .WORD   0

                                ;INITIALIZE STACK

 001365  104206  002260                 MOV #STACK,SP                   ;SET UP STACK POINTER
 001367                                 BR      START                   ; BRANCH OVER SUPPORT CODE
```

```
                                .SBTTL  UDA DM PROGRAM PARAMETERS

                                .LIST MEB
                        ;
                        ;       EQUATES
                        ;
                        ;       HIGHEST USABLE LOCATION OF UDA MEMORY + 1
                        ;
037777                  HIMEM   =       37777
                        ;
                        ;
                        ;       OFFSETS FOR FORMAT TRACK TABLE
                        ;
000000                  FT.BUF  =       0.                      ;BUFFER POINTER OFFSET
000001                  FT.HI   =       1.                      ;HI ORDER HEADER OFFSET
000002                  FT.LOW  =       2.                      ;LOW ORDER HEADER OFFSET
                        ;
                        ;
                        ;       OFFSETS FOR FORMAT TRACK BUFFER
                        ;
000000                  FB.DAT  =         0.                    ;FIRST DATA WORD OFFSET
000400                  FB.EDC  =       256.                    ;EDC WORD OFFSET
                        ;
                        ;
                        ;       OFFSETS FOR READ/WRITE I/O CHAIN TABLES
                        ;
000000                  RW.CPT  =       0.                      ;NEXT BUFFER POINTER OFFSET
000001                  RW.STAT =       RW.CPT+1.               ;STATUS
000002                  RW.BUF  =       RW.STAT+1.              ;POINTER TO DATA BUFFER
000003                  RW.LOW  =       RW.BUF+1.               ;HI ORDER EXPECTED HEADER
000004                  RW.HI   =       RW.LOW+1.               ;LOW ORDER EXPECTED HEADER
000005                  RW.CMD  =       RW.HI+1.                ;SDI COMMAND AND HEAD ADDRESS
000006                  RW.SDI  =       RW.CMD+1.               ;DUMMY SDI CONTROL BLOCK POINTER
000007                  RW.ANG  =       RW.SDI+1.               ;THETA FROM INDEX
                        ;
                        ;       CONSTANTS FOR READ AND WRITE XFC'S
                        ;
140000                  WSTOP   =       140000                  ; LAST ENTRY IN CHAIN FOR WRITE
040000                  WCONT   =       40000                   ; WRITE CONTINUE
100000                  RSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR READ
000000                  RCONT   =       0                       ; READ CONTINUE
100000                  FSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR FORMAT
122400                  WREAL   =       122400                  ; WRITE REAL TIME ECOMMAND
013400                  RREAL   =       13400                   ; READ REAL TIME COMMAND
010000                  ECCFLG  =       10000                   ; ECC ERROR IN BUFFER BIT
100000                  EOC     =       100000                  ; END OF CHAIN
040000                  BUFFLG  =       40000                   ; BUFFER FULL OR EMPTY FLAG
001750                  MAXSND  =       1000.
                        ;
                        ;
                        ;       HEADER CODES
                        ;
000000                  HD.LBN  =       000000                  ;GOOD LBN
060000                  HD.RBN  =       060000                  ;GOOD RBN, PERHAPS UNUSED
030000                  HD.REV  =       030000                  ;REVECTORED LBN
110000                  HD.BAD  =       110000                  ;BAD BLOCK
050000                  HD.PRV  =       050000                  ;PRIMARY REVECTORED BLOCK
```

```
     81          120000                  HD.XBN  =       120000                  ;XBN BLOCK
     82          140000                  HD.DBN  =       140000                  ;DBN BLOCK
     83
     84                                  ;       LEVEL 1 CODES
     85          070400                  MS.STR  =       70400                   ;MESSAGE START
     86          131000                  MS.END  =       131000                  ;MESSAGE END
     87          152000                  MS.CNT  =       152000                  ;MESSAGE CONTINUE
     88          107000                  SL.GRP  =       107000                  ;SELECT GROUP
     89
     90                                  ;       OFFSETS FOR DATA BUFFERS
     91
     92          000000                  BF.DAT  =         0.                    ;DATA
     93          000400                  BF.EDC  =       256.                    ;ERROR DETECTION CODE
     94          000401                  BF.ECC  =       257.                    ;LAST 17 ECC RESIDUES
     95
     96                                  ;       BUFFER AND READ/WRITE CHAIN LINK SIZES
     97
     98          000401                  WBUFLN  =       257.                    ; WRITE BUFFER SIZE
     99          000415                  RBUFLN  =       WBUFLN+12.              ; READ BUFFER SIZE
    100          000007                  LINKLN  =       7.                      ; LINK SIZE
    101
    102                                  ;       UNIT PARAMETER USED BY TEST3
    103
    104          000050                  U.SUBU  =       50
    105          000063                  U.UNUM  =       63
    106
    107                                  ;       FORMAT CHAIN SIZE
    108
    109          001375                  FOR.SZ  =       255.*3
```

```
                        ;       XFC DEFINITION EQUATES
                        ;
000000  BREAK   =       0.      ;BREAKPOINT XFC CODE
000001  FORMAT  =       1.      ;FORMAT TRACK XFC CODE
000002  XREAD   =       2.      ;READ N SECTORS XFC CODE
000003  XWRITE  =       3.      ;WRITE N SECTORS XFC CODE
000004  SEND    =       4.      ;SEND SDI COMMAND XFC CODE
000005  RCV     =       5.      ;RECEIVE SDI MESSAGE XFC CODE
000006  COMPARE =       6.      ;COMPARE DATA PATTERN TO BUFFER
000007  STATUS  =       7.      ;RETURN DRIVE STATUS XFC CODE
000010  ECHO    =       8.      ;ECHO DATA TO DRIVE XFC CODE
000011  DINIT   =       9.      ;DRIVE INITIALIZE XFC CODE
000012  WAITSI  =       10.     ;WAIT FOR SECTOR OR INDEX PULSE
000013  UREAD   =       11.     ;READ UNIBUS MEMORY XFC CODE
000014  UWRITE  =       12.     ;WRITE UNIBUS MEMORY XFC CODE
000015  ECC     =       13.     ;DO ECC ON BUFFER XFC CODE
000016  MRD     =       14.     ;SEND BUFFER TO MAINTENANCE READ COMMAND
000017  MWR     =       15.     ;GET BUFFER FROM MAINTENANCE WRITE COMMAND
000020  CVT     =       16.     ;CONVERT TO PHYSICAL ADDRESS XFC CODE
000021  EXIT    =       17.     ;TERMINATE DM PROGRAM XFC CODE
                        ;
                        ;       MEDIA TYPE
                        ;
126736  MOD512 = 126736
074161  MOD576 = 074161
                        ;
                        ;       GET STATUS OFFSETS
                        ;
000000  ST.UNT   =       0.      ;UNIT NUMBER
000000  ST.MSK   =       0.      ;SUBUNIT MASK
000001  ST.STA   =       1.      ;STATUS BYTE
000001  ST.MOD   =       1.      ;MODE BYTE
000002  ST.ERR   =       2.      ;ERROR BYTE
000002  ST.CON   =       2.      ;CONTROLLER BYTE
000002  ST.C     =       2.      ;C BITS
000003  ST.RTY   =       3.      ;RETRY COUNT/FAILURE CODE
                        ;
                        ;       STATUS BIT DEFINITIONS
                        ;
000010  ST.EL    =       10      ; LOGGABLE INFO IN EXTENDED STATUS
000200  ST.OA    =       200     ; ONLINE TO ANOTHER (SET IF DRIVE UNAVAILABLE)
000100  ST.RR    =       100     ; READJUSTMENT BIT (SET IF RECALIBRATION REQUIRED)
000040  ST.DR    =       40      ; DIAGNOSTIC REQUEST (SET IF DIAGNOSTIC REQUESTED)
000020  ST.SR    =       20      ; SPINDLE READY (SET IF SPINDLE READY)
000002  ST.PS    =       2       ; PORT SWITCH (SET IF PORT SWITCH IN)
000001  ST.RU    =       1       ; RUN/STOP SWITCH (SET IF RUN/STOP SWITCH IN)
000200  ST.FE    =       200     ; FATAL ERROR (SET IF FATAL ERROR OCCURRED)
000100  ST.RE    =       100     ; RETRIABLE ERROR (SET IF RETRIABLE ERROR OCCURRED)
000040  ST.PE    =       40      ; PROTOCOL ERROR (SET IF PROTOCOL ERROR OCCURRED)
000020  ST.DF    =       20      ; INITIALIZATION FAILURE (SET IF INIT FAILED)
000010  ST.WE    =       10      ; WRITE ENABLE (SET IF WRT ATTEMPTED ON PROT DISK)
002000  ST.FO    =       2000    ; FORMATTING (SET IF FORMATTING ENABLED)
001000  ST.DB    =       1000    ; DIAGNOSTIC CYLS (SET IF DIAG CYL ACCESS ENABLED)
000400  ST.S7    =       400     ; SECTOR SIZE (SET FOR 576 BYTE SECTORS)
```

```
                                ;       GET COMMON CHARACTERISTICS OFFSETS
                                ;
000000                          SHRTTO  =       0.              ;SHORT TIMEOUT <3:0>
000000                          SDIVER  =       0.              ;SDI VERSION <7:4>
000000                          XFERRT  =       0.              ;TRANSFER RATE <15:0>
000001                          LONGTO  =       1.              ;LONG TIMEOUT <3:0>
000001                          RETS    =       1.              ;RETRIES <7:4>
000001                          RCTCPS  =       1.              ;F/RCT COPIES <11:8>
000001                          SS      =       1.              ;SECTOR SIZE <15:15>
000002                          ERLEV   =       2.              ;ERROR RETRY LEVELS <7:0>
000002                          ECCRSH  =       2.              ;ECC THRESHOLD <15:8>
000003                          MICREV  =       3.              ;MICROCODE REVISION NUMBER <7:0>
000003                          HRDREV  =       3.              ;HARDWARE REVISION NUMBER <15:8>
000004                          DRVID   =       4.              ;UNIQUE DRIVE ID <47:0>
000007                          DRTYPE  =       7.              ;DRIVE TYPE IDENTIFIER <7:0>
000007                          REVS    =       7.              ;REVS/SECOND <15:8>
                                ;
                                ;       GET SUBUNIT CHARACTERISTICS OFFSETS
                                ;
                                ;THESE OFFSETS ARE CURRENTLY GIVEN AS FOLLOWING THE COMMON CHARACTERISTICS
                                ;
000013                          SUB     =       11.             ;OFFSET TO PUT SUBUNIT AFTER COMMON;
000000                          LBNCYL  =       0.              ;NUMBER OF CYLINDERS IN LBN AREA <31:0>
000001                          HICYL   =       1.              ;HI ORDER CYLINDER BITS <15:12>
000002                          GRPCYL  =       2.              ;GROUPS PER CYLINDER <7:0>
000002                          HILBN   =       2.              ;HI STARTING LBN <11:8>
000002                          HIXBN   =       2.              ;HI STARTING XBN <15:12>
000003                          TRKGRP  =       3.              ;TRACKS PER GROUP <7:0>
000003                          HIRBN   =       3.              ;HI STARTING RBN <11:8>
000003                          HIDBN   =       3.              ;HI STARTING DBN <15:12>
000004                          RBNTRK  =       4.              ;RBNS PER TRACK <6:0>
000004                          RM      =       4.              ;REMOVABLE MEDIA <7:7> 1=REMOVEABLE
000005                          DATPRE  =       5.              ;DATA PREAMBLE SIZE IN WORDS <7:0>
000005                          HDRPRE  =       5.              ;HEADER PREAMBLE SIZE IN WORDS <15:8>
000006                          MEDTYP  =       6.              ;MEDIA TYPE <31:0>
000010                          FCTSIZ  =       8.              ;FCT COPY SIZE <15:0>
000011                          LBNTRK  =       9.              ;LBNS PER TRACK <7:0>
000011                          GRPOFF  =       9.              ;GROUP OFFSET (SECTORS) <15:8>
000012                          LBNHST  =       10.             ;LBNS IN HOST AREA <31:0>
000014                          RCTCSZ  =       12.             ;RCT COPY SIZE <15:0>
000021                          XBNCYL  =       17.             ;CYLS IN XBN AREA <15:0>
000022                          DBNCYL  =       18.             ;CYLS IN DBN AREA <15:8>
                                ;
                                ;       UNIT CODES
                                ;
000001                          UNIT0   =       1.              ;UNIT ZERO CODE
000002                          UNIT1   =       2.              ;UNIT ONE CODE
000004                          UNIT2   =       4.              ;UNIT TWO CODE
000010                          UNIT3   =       8.              ;UNIT THREE CODE
                                ;
                                ;       BIT MASK DEFINITIONS
                                ;
                                ;
177400                          HIBYTE  =       177400          ;HIGH BYTE MASK
000377                          LOBYTE  =       000377          ;LOW BYTE MASK
007777                          HBHINB  =       7777            ;HI BYTE, HI NIBBLE MASK
170377                          HBLONB  =       170377          ;HI BYTE, LO NIBBLE MASK
```

```
                177417          LBHINB  =       177417          ;LO BYTE, HI NIBBLE MASK
                177760          LBLONB  =       177760          ;LO BYTE, LO NIBBLE MASK
                                ;
                000001          TIMEOUT =       1.              ;DRIVE TIMEOUT CODE
                000002          HEADER  =       2.              ;HEADER COMPARE FAILURE CODE
                000004          REVECT  =       4.              ;REVECTOR NEEDED CODE
                                ;
                000002          WRONG   =       2.              ;FIRST WORD NOT START FRAME CODE
                000004          FRAME   =       4.              ;FRAMING ERROR CODE
                000010          CHECK   =       8.              ;CHECKSUM ERROR CODE
                                ;
                000001          TOOBIG  =       1.              ;NUMBER OF WORDS EXCEEDS 7064
                000002          LOW     =       2.              ;DM BUFFER ADDRESS IS LESS THAN 714
                                ;
                                ;       BIT MASK
                                ;
                000001          LARGE   =       1.              ;BLOCK NUMBER TOO LARGE
                000002          OVERFL  =       2.              ;SECTOR NUMBER LARGER THAN 16 BITS
```

```
                        ;MAINTANENCE READ/WRITE REQUEST NUMBERS
                        ;
060000                  T1MSIZ  =       0.+60000        ;GET FREE MEMORY PARAMETERS
060001                  T2DLL   =       1.+60000        ;DOWNLINE LOAD DRIVE DIAGNOSTIC
060002                  T2CMD   =       2.+60000        ;MANUAL INTERVENTION TEST 2 PROTOCOL
060003                  T4MPRM  =       3.+60000        ;GET MASTER PARAMETERS FROM SW QUESTIONS
060004                  T4UPRM  =       4.+60000        ;GET UNIT PARAMETERS FROM HW QUESTIONS
060005                  T4BB1   =       5.+60000        ;GET BAD BLOCKS (1 THRU 14)
060006                  T4BB2   =       6.+60000        ;GET REST OF BAD BLOCKS (15 AND 16)
060007                  T4SOFT  =       7.+60000        ;ADD TO SOFT ERROR AND ECC COUNT
060010                  T4SEEK  =       8.+60000        ;ADD 1 TO SEEK COUNT
060011                  T4MXFR  =       9.+60000        ;ADD TO MEGABITS READ AND WRITTEN
060012                  UTOTST  =       10.+60000       ;GET UNITS TO TEST
060013                  ERRMES  =       11.+60000       ;PRINT ERROR MESSAGE
060014                  ERRMC   =       12.+60000       ;TEST 4  RROR REPORTING
060015                  MESSAG  =       13.+60000       ;INFORMA ION MESSAGE
060016                  DONE    =       14.+60000       ;MARK DM PROGRAM AS NO LONGER RUNNING
                        ;
                        ;       OTHER BIT DEFINITIIONS
                        ;
000001                  RCVRDY  =       1               ; RECIEVER READY 1 = READY
000002                  ATTN    =       2               ; ATTENTION BIT FOR RETURN DRIVE SIGNALS XFC
000004                  DCLOCK  =       4               ; TRANSMIT ERROR
000100                  AVAIL   =       100             ; AVAILABLE 1 = AVAILABLE
000400                  RCVERR  =       400             ; RECIEVER ERROR
100000                  RWRDY   =       100000          ; IF SET, UDA IS ABLE TO READ AND/OR WRITE TO DRIVE
                        ;
                        ;       SDI COMMANDS AND RESPONSES
                        ;
000204                  DISCON  =       204             ; DISCONNECT DRIVE
000006                  ERECOV  =       6               ; ERROR RECOVERY
000201                  CHGMOD  =       201             ; CHANGE MODE
000213                  DRVONL  =       213             ; DRIVE ONLINE
000014                  DRVRUN  =       14              ; DRIVE RUN
000005                  DRVCLR  =       5               ; DRIVE CLEAR OPCODE
000207                  GETCHR  =       207             ; GET CHARACTERISTICS
000210                  GETSUB  =       210             ; GET SUBUNIT CHARACTERISTICS
000011                  GETSTA  =       11              ; GET STATUS
000216                  IRECLB  =       216             ; RECALIBRATE
000012                  INSEEK  =       12              ; INITIATE SEEK
000176                  COMPLT  =       176             ; SUCCESSFUL COMPLETION
000175                  UNSSUC  =       175             ; UNSUCCESSFUL COMPLETION
000170                  CHRRES  =       170             ; GET CHARACTERISTICS RESPONSE
000167                  SBCRES  =       167             ; GET SUBUNIT CHARACTERISTICS RESPONSE
000366                  STSRES  =       366             ; GET STATUS RESPONSE
000350                  ECHOC   =       350             ; DIAGNOSTIC ECHO COMMAND AND RESPONSE
                        ;
                        ;       ERROR CODES
                        ;
000000                  FTLSYS  =           0           ; SYSTEM FATAL ERROR
040000                  FTLDEV  =       40000           ; DEVICE FATAL
100000                  ERHARD  =      100000           ; HARD ERROR
140000                  ERSOFT  =      140000           ; SOFT ERROR
040000                  C2HARD  =      <ERHARD&^CERSOFT>!<ERSOFT&^CERHARD>     ; CHANGE SOFT TO HARD ERROR
```

```
                                .SBTTL  TEST 4 SPECIFIC INFORMATION
                        ;
                        ;       TEST 4 SPECIFIC INFORMATION
                        ;
                        ;
                        ;       CONSTANTS
                        ;
000377                  SCTWRD  =       255.                    ; NUMBER OF WORDS IN SECTOR TO FILL
000105                  INTEDC  =       69.                     ; INITIAL EDC VALUE
```

```
                        .SBTTL  MACRO DEFINITIONS
                ;
                ;       MESSAGE CONTROL TABLE MACRO
                ;
                        .MACRO MSG CMDBUF,CMDSZ,RPLBUF,RPLSZ,SUCCOM
                        .WORD CMDBUF                    ;ADDRESS OF COMMAND
                        .WORD CMDSZ                     ;SIZE OF COMMAND IN BYTES
                        .WORD RPLBUF                    ;ADDRESS OF REPLY
                        .WORD RPLSZ                     ;SIZE OF REPLY IN WORDS
                        .IF NB NUMBER
                        .WORD SUCCOM                    ; SUCCESSFUL COMPLETION CODE
                        .ENDC
                        .ENDM
```

MACRO DEFINITIONS

```
      1                                 .MACRO  BCS     LAB..,?B
      2                                                                 BCC     B
      3                                                                 BR      LAB..
      4                           B:
      5                                 .ENDM
```

```
                        ;       PUSH REGISTER MACRO
                        ;
                                .MACRO PUSH R9
                                .IRP X,<R9>
                                                        MOV X,-(SP)
                                .ENDR
                                .ENDM
                        ;
                        ;       POP REGISTER MACRO
                        ;
                                .MACRO POP R9
                                .IRP X,<R9>
                                                        MOV (SP)+,X
                                .ENDR
                                .ENDM
```

```
                        ;ERROR MACROS
                        ;THESE MACROS ARE CALLED TO REPORT ERRORS TO THE HOST PROGRAM.
                        ;THE MACRO NAMES ARE : ERRSF, ERRDF, ERRHRD, ERRSFT. EACH RESULTS IN THE HOST
                        ;BEING REQUESTED TO REPORT THE ERROR.
                        ;ARGUMENTS:     1 (MS$) MESSAGE POINTER
                        ;               2 (P1$) PARAMETER #1
                        ;               3 (P2$) PARAMETER #2
                        ;               4 (P3$) PARAMETER #3
                        ;               5 (P4$) PARAMETER #4
                        ;               6 (P5$) PARAMETER #5
                        ;               7 (P6$) PARAMETER #6
                        ;               8 (P7$) PARAMETER #7
                        ;               9 (P8$) PARAMETER #8
                        ;
                        ;THE MESSAGE POINTER MUST POINT TO AN ADDRESS IN THE OVERLAY 'MS'' IMMEDIATELY
                        ;FOLLOWING THE MAIN CODE. ANY ADDRESS MODE MAY BE USED (E.G. #MS1, @R2).
                        ;THE ADDRESS MUST CONTAIN AN ASCII FORMAT STRING TO DETERMINE THE MESSAGE
                        ;TO PRINT.
                        ;THE PARAMETER ARGUMENTS ARE OPTIONAL. THEY SHOULD BE SUPPLIED ONLY WHEN
                        ;THERE IS DATA TO BE PASSED TO THE HOST THAT WILL BE USED IN PRINTING THE
                        ;MESSAGE. THESE PARAMETER ARGUMENTS ARE THE ADDRESS OF DATA TO BE PASSED
                        ;USING ANY ADDRESSING MODE DESIRED.
                        ;ALL REGISTERS ARE RETURNED UNCHANGED. IT SHOULD BE NOTED THAT ARGUMENTS
                        ;CONTAINING SOMETHING OTHER THAN A REGISTER NAME (E.G. #100 OR MEMADR)
                        ;ASSEMBLE TO INSTRUCTIONS THAT SAVE AND RESTORE A REGISTER ON THE STACK.

                                .MACRO ERRSF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ FTLSYS,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRDF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ FTLDEV,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRHRD MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NLIST
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ ERHARD,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .LIST
                                .ENDM

                                .MACRO ERRSFT MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ ERSOFT,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM
```

```
                        ;THE FOLLOWING MACRO ACTUALLY PROCESSES THE ERROR CALL TO THE HOST PROGRAM

                .MACRO ERROR$ ET$,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,ERRN$
                  .RADIX 8
                  PRMS=ARGS$-1
                  .IF LT,<PRMS>
                    .ERROR;NOT ENOUGH ARGUMENTS IN ERROR CALL
                  .ENDC
                  REGS$=-1
                  .IIF GE,<PRMS-8.>,PARGS. P8$
                  .IIF GE,<PRMS-7.>,PARGS. P7$
                  .IIF GE,<PRMS-6.>,PARGS. P6$
                  .IIF GE,<PRMS-5.>,PARGS. P5$
                  .IIF GE,<PRMS-4.>,PARGS. P4$
                  .IIF GE,<PRMS-3.>,PARGS. P3$
                  .IIF GE,<PRMS-2.>,PARGS. P2$
                  .IIF GE,<PRMS-1.>,PARGS. P1$
                  .IF GE REGS$
                    RSTR$ \REGS$
                  .ENDC
                  .RADIX 10
                  .LIST
                                                                CALL RERROR     ;ERROR # ERRN$'.
                  .NLIST
                  .RADIX 8
                  .LIST

                                                                .WORD ET$+ERRN
                                                                .WORD <PRMS*10000>+MS$

                  .NLIST
                  ERRN=ERRN+1
                .ENDM

                .MACRO PARGS.,ADDR$
                  .NTYPE PTYPE$,ADDR$
                  .IF EQ,<PTYPE$&70>
                    .IIF EQ,<PTYPE$&7>-REGS$,RSTR$ \REGS$
                    .LIST
                                                                MOV ADDR$,-(SP)
                    .NLIST
                  .IFF
                    .IF EQ,<PTYPE$&7>-1         ;PICK A REGISTER TO USE
                      REGU$=2                   ;SELECT R2 IF R1 IS USED IN PARAMETER FETCH
                    .IFF
                      REGU$=1                   ;OTHERWISE USE R1
                    .ENDC
                    .IF NE,<REGU$-REGS$>        ;IF REGISTER NOT ALREADY SAVED
                      .IF GE,REGS$
                        RSTR$ \REGS$            ;RESTORE CURRENT SAVED REGISTER
                      .ENDC
                      SAVR$ \REGU$              ;THEN SAVE SELECTED REGISTER
                    .ENDC
                    GETP$ \REGS$,ADDR$
                  .ENDC
                .ENDM
```

MACRO DEFINITIONS

```
                        .MACRO SAVR$ REGN
                          .LIST
                                                        MOV R'REGN,SAVREG
                          .NLIST
                          REGS$=REGN
                        .ENDM

                        .MACRO RSTR$ REGN
                          .LIST
                                                        MOV SAVREG,R'REGN
                          .NLIST
                          REGS$=-1
                        .ENDM

                        .MACRO GETP$ REGN,ADDR$
                          .LIST
                                                        MOV ADDR$,R'REGN
                                                        MOV R'REGN,-(SP)
                          .NLIST
                        .ENDM
```

```
                        ;
                        ;       PRIMARY ERROR REPORTING (TEST 4)
                        ;
                        .MACRO  SOFTER  NUM,ARGS
                        ERROR   ERSOFT,NUM,<ARGS>
                                                        MOV     #ERRMES,OUT.RQ
                        .ENDM

                        .MACRO  REPSFT  SFTFLG,ECCFG,SEKFLG
                        .IF     NB,SFTFLG
                                                        MOV     #1,OUT.02
                        .IFF
                                                        CLR     OUT.02
                        .ENDC
                        .IF     NB,ECCFG
                                                        MOV     #1,OUT.03
                        .IFF
                                                        CLR     OUT.03
                        .ENDC
                        .IF     NB,SEKFLG
                                                        MOV     #1,OUT.04
                        .IFF
                                                        CLR     OUT.04
                        .ENDC
                                                        PUSH    R0
                                                        MOV     U.UNUM(R5),R0
                                                        ADD     U.SUBU(R5),R0
                                                        MOV     R0,OUT.01
                                                        MOV     #T4SOFT,R0
                                                        CALL    HOSTRQ
                                                        POP     R0
                        .ENDM

                        .MACRO  HARDER  NUM,ARGS
                        ERROR   ERHARD,NUM,<ARGS>
                                                        MOV     #ERRMES,OUT.RQ
                        .ENDM

                        .MACRO  DEVFTL  NUM,ARGS
                        ERROR   FTLDEV,NUM,<ARGS>
                                                        MOV     #ERRMES,OUT.RQ
                        .ENDM

                        .MACRO  SYSFTL  NUM,ARGS
                        ERROR   FTLSYS,NUM,<ARGS>
                                                        MOV     #ERRMES,OUT.RQ
                        .ENDM

                        .MACRO  ERROR   TYPE,NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       5
                        MOVMSG  #MS'NUM,4
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
```

```
                                        .ENDC
                                                                        MOV     LUNIT,OUT.03
                                                                        MOV     #NUM!TYPE+3000.,R2
                                                                        MOV     R2,OUT.02
                                                                        MOV     #.,OUT.01
                                        .RADIX
                                        .ENDM

                                        .MACRO  CERROR  STNUM,ARGS
                                        .RADIX  10
                                NUMPTR  =       STNUM
                                        .IRP    X,<ARGS>
                                        MOVMSG  X,\NUMPTR
                                NUMPTR  =       NUMPTR + 1
                                        .ENDR
                                        .RADIX
                                        .ENDM

                                        .MACRO  ERRORC  ARGS
                                        .RADIX  10
                                        .IRP    X,<ARGS>
                                        MOVMSG  X,\NUMPTR
                                NUMPTR  =       NUMPTR + 1
                                        .ENDR
                                        .RADIX
                                        .ENDM

                                        .MACRO  MOVMSG  ARG,INDX

                                        .IF     LT,INDX-10
                                                                        MOV     ARG,OUT.0'INDX
                                        .IFF
                                                                        MOV     ARG,OUT.'INDX
                                        .ENDC
                                        .ENDM

                                        .MACRO  ENDERR  POS
                                        .RADIX  10
                                        .IF     NB,POS
                                        NDERR   POS
                                        .IFF
                                        NDERR   \NUMPTR
                                        .ENDC
                                        .RADIX
                                        .ENDM

                                        .MACRO  NDERR   POS
                                        .IF     NE,POS
                                        MOVMSG  #SER22,POS
                                                                        MOV     #POS+1,ERRPOS   ; SET THE POSITION
                                        .IFF
                                                                        CLR     ERRPOS          ; CLEAR THE POSITION
                                        .ENDC
                                        .ENDM
                                ;
                                ;       MESSAGE REPORTING MACRO
                                ;
```

```
                        .MACRO  MSSG    NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       3
                        MOVMSG  #MS'NUM,2
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC
                                                        PUSH    <R0,R1>
                                                        MOV     LUNIT,OUT.01
                                                        MOV     #MESSAG,R0
                                                        CALL    HOSTRQ
                                                        POP     <R1,R0>
                        .RADIX
                        .ENDM
                    ;
                    ;
                        .MACRO  MSSGE   NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       3
                        MOVMSG  #'NUM,2
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC
                                                        PUSH    <R0,R1>
                                                        MOV     #MESSAG,R0
                                                        CALL    HOSTRQ
                                                        POP     <R1,R0>
                        .RADIX
                        .ENDM
```

MACRO DEFINITIONS

```
      1                                 ;ASSUME MACRO
      2                                         .MACRO  ASSUME  P1,P2
      3                                         .IF     NE,P1-P2
      4                                         .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
      5                                         .ENDC
      6                                         .ENDM
```

MACRO DEFINITIONS

```
                        ;RETURN DRIVE STATUS MACRO WITH ERROR REPORTING

                                .MACRO  DSTAT,LAB$,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   ME
                                .LIST
                                CALL    RDSTAT                  ; GET DRIVE STATUS
                                BIT     #10000,R1               ; SEE IF ANY ERRORS
                                BEQ     2$                      ; IF NO ERROR, BRANCH
                                BIT     #4000,R1                ; SEE IF XMIT ERROR
                                BEQ     1$                      ; IF SO, BRANCH
                                .NLIST  ME
                                .LIST   MEB
                                HARDER  E1                      ; REPORT INVALID STATUS ERROR
                                .NLIST  MEB
                                .LIST   ME
                                BR      LAB$                    ; BRANCH TO DONE
                        1$:
                                .NLIST  ME
                                .LIST   MEB
                                HARDER  E2                      ; REPORT XMIT ERROR
                                .NLIST  MEB
                                .LIST   ME
                                BR      LAB$                    ; BRANCH TO DONE
                        2$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

MACRO DEFINITIONS

```
      1                                ;
      2                                ;       XOR THE CONTENTS OF TWO REGISTERS
      3                                ;
      4                                        .MACRO  RXOR    REG1,REG2
      5                                        MOV     REG2,-(SP)              ; SAVE REGISTER REG2
      6                                        BIC     REG1,REG2               ; CLEAR COORESPONDING BITS IN REG2
      7                                        BIC     (SP)+,REG1              ; CLEAR COORESPONDING BITS IN REG1
      8                                        BIS     REG1,REG2               ; OR WHAT'S LEFT
      9                                        .ENDM
```

MACRO DEFINITIONS

```
                        ;
                        ;       CONVERT .BLKW CALLS TO ACTUAL WORDS GENERATED
                        ;
                        .MACRO  .BLKW   COUNT
                        .NLIST
                        .REPT   COUNT
                        .WORD   0
                        .ENDR
                        .LIST
                        .ENDM
```

```
                        ;       SDI INTERCHANGE WITH DRIVE WITH ERROR REPORTING
                        
                                .MACRO  TALKX   ERRLAB,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   ME
                                .LIST
                                CALL    TALK                    ; INITIATE SDI INTERCHANGE
                                TST     R2                      ; SEE IF ERROR OCCURRED
                                BEQ     12$                     ; IF NOT, BRANCH
                                BPL     11$                     ; IF SO, BRANCH
                                .NLIST  ME
                                .LIST   MEB
                                HARDER  E1                      ;SEND COMMAND ERROR
                                .NLIST  MEB
                                .LIST   ME
                                BR      ERRLAB
                        11$:
                                .NLIST  ME
                                .LIST   MEB
                                HARDER  E2                      ;RECEIVE COMMAND ERROR
                                .NLIST  MEB
                                .LIST   ME
                                BR      ERRLAB
                        12$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

```
                                        .SBTTL  MACRO FOR OVERLAY TABLE
                                        .MACRO  DFOVLY  LABL$,ONAME,SAREA,EAREA
                                        .WORD   0
                                        .WORD   OVL.'ONAME'*4
                                        .IF     NE,OCNT$-LABL$
                                        .ERROR  ; OVERLAY NUMBER AND POSITION IN TABLE DO NOT MATCH
                                        .ENDC
                                OCNT$   =       OCNT$+1
                                        .ENDM

                                        .MACRO  MESSAGES
                                        .SBTTL  MESSAGES
                                ;MESSAGE STORAGE OVERLAY


                                        DMOVLY MS,0
                                        .NLIST BEX

                                MS1:    .ASCII\''TIME-OUT ON SEND'N\
                                        .ASCII\R1R1\
                                        .BYTE 0
                                MS2:    .ASCII\''TIME-OUT ON RECEIVE'N\
                                        .ASCII\R1R1\
                                        .BYTE 0
                                MS3:    .ASCII\''FIRST WORD RECEIVED WAS NOT A START FRAME'N\
                                        .ASCII\R1R1\
                                        .BYTE 0
                                MS4:    .ASCII\''FRAMING ERROR ON LEVEL 0 RESPONSE'N\
                                        .ASCII\R1R1\
                                        .BYTE 0
                                MS5:    .ASCII\''CHECKSUM ERROR ON LEVEL 0 RESPONSE'N\
                                        .ASCII\R1R1\
                                        .BYTE 0
                                MS6:    .ASCII\''RESPONSE LONGER THAN EXPECTED'N\
                                        .ASCII\R1R1\
                                        .BYTE 0
                                MS7:    .ASCII\''CODE FROM RECEIVE WAS UNINTELLIGIBLE FROM SUBSYSTEM = 'H16N\
                                        .ASCII\R1R1\
                                        .BYTE 0
                                MS8:    .ASCII\''COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
                                        .ASCII\R1\
                                        .ASCII\''  EXPECTED RESPONSE  'H8N\
                                        .ASCII\''  ACTUAL RESPONSE    'H8N\
                                        .ASCII\R1\
                                        .BYTE 0
                                MS9:    .ASCII\''DRIVE NOT ASSERTING RECEIVER READY IN DRIVE STATE'N\
                                        .BYTE 0
                                MS10:   .ASCII\''FAILED TO RECEIVE VALID DRIVE STATE'NR1\
                                        .BYTE 0
                                MS11:   .ASCII\''CANNOT RECEIVE DRIVE STATE FROM DRIVE'N\
                                        .ASCII\''CHECK IF DRIVE IS POWERED ON.'NR1\
                                        .BYTE 0
                                MS12:   .ASCII\''DRIVE STATE RECEIVED HAS BAD PARITY'NR1\
                                        .BYTE 0
                                MS13:   .ASCII\''NO VALID STATE FROM DRIVE'NR1\
                                        .BYTE 0
                                MS14:   .ASCII  \''SUBUNIT CHARACTERISTICS SAY THERE ARE ZERO READ ONLY GROUPS'N\
```

```
                                        .ASCII  \'IN THE DIAGNOSTIC AREA'N\
                                        .BYTE   0
                                MS15:   .ASCII  \'SUBUNIT CHARACTERISTICS SAY THERE ARE LESS THAN 1 READ/WRITE'N\
                                        .ASCII  \'GROUPS IN THE DIAGNOSTIC AREA'N\
                                        .BYTE   0
                                MS16:   .ASCII\'NEITHER R/W READY NOR ATTENTION SET AFTER RECALIBRATE COMMAND'NR1\
                                        .BYTE 0
                                MS17:   .ASCII\'SUBUNIT CHARACTERISTICS SAY LESS THAN 1 DIAGNOSTIC CYLINDER'N\
                                        .BYTE 0
                                MS18:   .ASCII\'READ/WRITE READY DROPPED BEFORE FORMAT OPERATION'N\
                                        .BYTE 0
                                MS19:   .ASCII\'FORMAT OPERATION REPORTED TIME-OUT FAILURE'N\
                                        .ASCII\'  CYLINDER 'D28'.   GROUP  'D8'.  TRACK 'D8'.'N\
                                        .BYTE 0
                                MS20:   .ASCII\'AFTER RECAL, ERROR BITS WERE SET'NR1\
                                        .BYTE 0
                                MS21:   .ASCII\N'LOGGABLE INFORMATION AFTER RECAL'NR1\
                                        .BYTE 0
                                MS22:   .ASCII\'READ/WRITE READY DROPPED BEFORE WRITE OPERATION'N\
                                        .BYTE 0
                                MS23:   .ASCII\'COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
                                        .ASCII\'WRITE OPERATION REPORTED FAILURE -- ERROR CODE 'D8' OCTAL.'N\
                                        .ASCII\'DBN 'D24'.  CYLINDER 'D28'.  GROUP 'D8'.  TRACK 'D8'.'N\
                                        .BYTE 0
                                MS24:   .ASCII\'READ/WRITE READY DROPPED BEFORE READ OPERATION'N\
                                        .BYTE 0
                                MS25:   .ASCII\'COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
                                        .ASCII\'READ OPERATION REPORTED FAILURE -- ERROR CODE 'D8' OCTAL.'N\
                                        .ASCII\'DBN 'D24'.  CYLINDER 'D28'.  GROUP 'D8'.  TRACK 'D8'.'N\
                                        .BYTE 0
                                MS26:   .ASCII\'COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
                                        .ASCII\'DATA COMPARE FAILURE ON WORD 'D16'.'N\
                                        .ASCII\'EXPECTED DATA  'H16N\
                                        .ASCII\'ACTUAL DATA    'H16N\
                                        .ASCII\'CYLINDER 'D28'.  GROUP 'D8'.  TRACK 'D8'.'N\
                                        .BYTE 0
                                MS27:   .ASCII\'SEEK COMPLETE TIME-OUT -- READ/WRITE READY DID NOT SET.'N\
                                        .ASCII\'SEEK WAS TO CYLINDER 'D28'.  GROUP 'D8'.'N\
                                        .BYTE 0
                                MS28:   .ASCII\'NO BLOCK ON THIS TRACK CAN BE READ. LAST BLOCK TRIED:'N\
                                        .ASCII\A1'BN 'D24'.  CYLINDER 'D28'.  GROUP 'D8'.  TRACK 'D8'.'N\
                                        .BYTE 0
                                MS29:   .ASCII  \'AVAILABLE WAS NOT ASSERTED AFTER DISCONNECT'N\
                                        .ASCII  \'  STATE RECEIVED  'H16N\
                                        .BYTE   0
                                MS30:   .ASCII\'INVALID COMMAND 'H16' WAS SUCCESSFUL'N\
                                        .BYTE   0
                                MS31:   .ASCII\'COMMAND WITH 'R1' LENGTH = 'D8' WAS SUCCESSFUL'N\
                                        .BYTE   0
                                MS32:   .ASCII\'UNIT DID NOT REPORT TRANSMITTION ERROR'NR1\
                                        .BYTE   0
                                MS33:   .ASCII\'UNIT ACCEPTED AN INVALID GROUP NUMBER FROM GROUP SELECT LEVEL 1'N\
                                        .BYTE   0
                                MS34:   .ASCII\'UNABLE TO CORRECTLY READ OVERLAY 'D3NR1\
                                        .BYTE   0
                                MS35:   .ASCII\'SUCCESSFULLY WROTE IN DBN AREA WHEN DRIVE WAS WRITE PROTECTED'N\
                                        .BYTE   0
```

```
                                MS36:   .ASCII  \'DRIVE IS NOT PROPERLY FORMATTED.'NR1\
                                        .BYTE   0
                                MS37:   .ASCII  \'DRIVE IS FORMATTED IN 576 BYTE MODE.'N\
                                        .ASCII  \'TO RUN WITH A UDA, THIS DRIVE NEEDS TO BE FORMATTED '\
                                        .ASCII  \'IN 512 BYTE MODE.'N\
                                        .BYTE   0
                                MS38:   .ASCII  \'NO COPY OF THE FCT COULD BE READ.'NR1\
                                        .BYTE   0
                                SER36:  .ASCII  \'UDA WILL SPIN DOWN THIS DRIVE IF USED IN NORMAL SYSTEM OPERATION.'N\
                                        .ASCII  \'THIS DRIVE NEEDS TO BE FORMATTED.'\
                                        .BYTE   0
                                SER39:  .ASCII\'THIS UDA AND ALL DRIVES ATTACHED WILL BE REMOVED FROM TESTING'N\
                                        .BYTE   0
                                SER00:  .ASCII\'COMMAND WAS 'R1\
                                        .BYTE   0
                                MS.ONL: .ASCII\'ONLINE'N\
                                        .BYTE   0
                                MS.CLR: .ASCII\'DRIVE CLEAR'N\
                                        .BYTE   0
                                MS.DIS: .ASCII\'DISCONNECT'N\
                                        .BYTE   0
                                MS.GCR: .ASCII\'GET COMMON CHARACTERISTICS'N\
                                        .BYTE   0
                                MS.SCR: .ASCII\'GET SUBUNIT CHARACTERISTICS'N\
                                        .BYTE   0
                                MS.GST: .ASCII\'GET STATUS'N\
                                        .BYTE   0
                                MS.MOD: .ASCII\'CHANGE MODE'N\
                                        .BYTE   0
                                MS.SEK: .ASCII\'SEEK'N\
                                        .BYTE   0
                                MS.INR: .ASCII\'INITIATE RECALIBRATE'N\
                                        .BYTE   0
                                MS.RUN: .ASCII\'SPIN UP'N\
                                        .BYTE   0
                                MS2000: .ASCII  \'UNABLE FIND REQUESTED DRIVE FOR TESTING'N\
                                        .ASCII  \'THE FOLLOWING IS VISIBLE ON THE PORTS'N\
                                        .ASCII  \'UDA PORT 0 -- 'R1\
                                        .ASCII  \'UDA PORT 1 -- 'R1\
                                        .ASCII  \'UDA PORT 2 -- 'R1\
                                        .ASCII  \'UDA PORT 3 -- 'R1\
                                        .BYTE   0
                                SER10:  .ASCII  \'NO DRIVE ATTACHED'N\
                                        .BYTE   0
                                SER11:  .ASCII  \'RCVR RDY NEVER ASSERTED'N\
                                        .BYTE   0
                                SER12:  .ASCII  \'TIMEOUT OF SEND'N\
                                        .BYTE   0
                                SER13:  .ASCII  \'TIMEOUT OF RECEIVE'N\
                                        .BYTE   0
                                SER14:  .ASCII  \'FIRST WORD RECEIVED WAS NOT START FRAME'N\
                                        .BYTE   0
                                SER15:  .ASCII  \'FRAMING ERROR ON LEVEL 0 RECEIVE'N\
                                        .BYTE   0
                                SER16:  .ASCII  \'CHECKSUM ERROR ON LEVEL 0 RECEIVE'N\
                                        .BYTE   0
                                SER17:  .ASCII  \'RESPONSE LONGER THAN EXPECTED FOR GET STATUS CMD'N\
```

```
                                        .BYTE   0
                                SER18:  .ASCII  \''DRIVE ''R1\
                                        .BYTE   0
                                SER18D: .ASCII  \D6'', ''\
                                SER18C: .ASCII  \D6'', ''\
                                SER18B: .ASCII  \D6'', ''\
                                SER18A: .ASCII  \D6N\
                                        .BYTE   0
                                SER22:  .ASCII\''REAL TIME STATE ''H16N\
                                        .ASCII\''STATUS (R TO L): ''H16S2H16S2H16S2H16S2H16S2H16S2H16N\
                                        .BYTE   0
                                SER23:  .ASCII\''REAL TIME STATE ''H16N\
                                        .BYTE   0
                                SER40:  .ASCII  \''DRIVE NOT AVAILABLE TO THIS UDA''N\
                                        .BYTE   0
                                SER41:  .ASCII  \''DRIVE NOT SPINABLE''N\
                                        .BYTE   0
                                SER50:  .ASCII\''COMMAND''\
                                        .BYTE   0
                                SER51:  .ASCII\''RESPONSE''\
                                        .BYTE   0
                                SER52:  .ASCII\''WHEN A CONTINUE OR END FRAME DID NOT FOLLOW A START FRAME''N\
                                        .BYTE   0
                                SER53:  .ASCII\''WHEN AN END FRAME WAS SENT WITH NO START FRAME''N\
                                        .BYTE   0
                                SER54:  .ASCII\''WHEN AN END FRAME WITH A BAD CHECKSUM WAS SENT''N\
                                        .BYTE   0
                                SER55:  .ASCII\''WHEN A CONTINUE FRAME WAS SENT WITH NO START FRAME''N\
                                        .BYTE   0
                                SER56:  .ASCII\''WHEN TWO CONSECUTIVE START FRAMES WERE SENT''N\
                                        .BYTE   0
                                SER57:  .ASCII\''WHEN AN END FRAME WAS SENT AFTER A START FRAME TIMED OUT''N\
                                        .BYTE   0

                                OVL.MS  = .
                                        .ENDM
```

```
                                        .SBTTL  RTDS   - REAL TIME DRIVE STATE ROUTINE WITH ERROR REPORTING (TEST 4)
                                ;
                                ;       DIFF BETWEEN SEND + RECEIVE ERRORS, REPORT HERE
                                ;
001371                          RTDS:   CALL    RTDSL           ; GET REAL TIME DRIVE STATE
001371  020000  001422                  ^020000,RTDSL
001373  115002                          TST     R2              ; SEE IF ERROR OCCURRED
001374                                  BEQ     1$              ; IF NOT, BRANCH
001374  010000  001420                  ^010000,1$
001376                                  DEVFTL  13              ; REPORT DEVICE FATAL ERROR
001376  104200  000461  001524                                          MOV     #MS13,OUT.04
001401  104300  002212  001523                                          MOV     LUNIT,OUT.03
001404  104202  045705                                                  MOV     #13!FTLDEV+3000.,R2
001406  104020  001522                                                  MOV     R2,OUT.02
001410  104200  001410  001521                                          MOV     #.,OUT.01
001413  104200  060013  001520                                          MOV     #ERRMES,OUT.RQ
001416                                  ENDERR  0
001416  114000  003626                                                  CLR     ERRPOS          ; CLEAR THE POSITION
001420                          1$:     RETURN
001420  000000  000000                  ^00,0

                                        .SBTTL  RTDSL - CALL RDSTAT
001422                          RTDSL:
001422  104302  003356                  MOV     SDI,R2                  ; GET INTERCONNECT CODE
001424                                  CALL    RDSTAT
001424  020000  001432                  ^020000,RDSTAT
001426  103201  077674                  BIC     #077674,R1              ; CLEAR UNUSED BITS
001430                                  RETURN
001430  000000  000000                  ^00,0

                                        .SBTTL  RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
001432                          RDSTAT:
                                ;
                                ;RETURN DRIVE STATUS
                                ;STATUS RETURNED IN DM REGISTER 1
                                ;
                                ;       INPUT R2 MUST HAVE INTERCONNECT CODE
                                ;
001432                                  PUSH    <R3,R0>                 ; SAVE R3 AND R0
001432  100463                                                                  MOV R3,-(SP)
001433  100467                                                                  MOV R0,-(SP)
001434  104203  024000                  MOV     #24000,R3               ; ERROR COUNT
001436  060007                  1$:     XFC     STATUS                  ; GET DRIVE'S STATUS
001437  102201  000004                  BIT     #DCLOCK,R1              ; SE IF DRIVE CLOCK IS PRESENT
001441                                  BEQ     3$                      ; IF NOT, BRANCH
001441  010000  001457                  ^010000,3$
001443  102201  000400                  BIT     #RCVERR,R1              ; RECIEVER ERRORS
001445                                  BEQ     2$                      ; IF NOT VALID, BRANCH
001445  010000  001456                  ^010000,2$
001447  117403                          DEC     R3                      ; DECREMENT ERROR COUNT
001450                                  BNE     1$                      ; IF ERROR COUNT NON-ZERO, BRANCH
001450  050000  001436                  ^050000,1$
001452  104202  177777                  MOV     #177777,R2              ; MARK AS RECEIVE ERROR
001454                                  BR      3$
001454  000000  001457                  ^00,3$
001456  114002                  2$:     CLR     R2                      ; NO ERRORS
001457                          3$:     POP     <R0,R3>                 ; RESTORE R0, R3
```

```
        001457  104267                                                  MOV (SP)+,R0
        001460  104263                                                  MOV (SP)+,R3
     40 001461                          RETURN                          ; RETURN TO CALLING MODULE
        001461  000000  000000          ^O0,0
```

HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED,

```
                                        .SBTTL  HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
001463                          HOSTRQ:
                                ;
                                ;SEND REQUEST BUFFER TO HOST AND WAIT FOR RESPONSE.
                                ;CLEAR ARGUMENT AREA OF OUT BUFFER IN PREPARATION
                                ;FOR NEXT HOSTRQ CALL.
                                ;
                                ;INPUTS:
                                ;       R0 - HOST REQUEST NUMBER
                                ;       OUT BUFFER LOADED WITH DATA
                                ;
001463                          PUSH    <R0,R1,R2>
001463  100467                                                                  MOV R0,-(SP)
001464  100461                                                                  MOV R1,-(SP)
001465  100462                                                                  MOV R2,-(SP)
001466  104300  002212  001523  MOV     LUNIT,OUT.03
001471  104070  001520          MOV     R0,OUT.RQ               ; STORE REQUEST NUMBER IN BUFFER
001473  104207  001520  SNDAGN: MOV     #OUT.RQ,R0              ; SEND BUFFER TO HOST
001475  104201  000043          MOV     #BUFSIZ,R1
001477  060016                  XFC     MRD
001500  115001                  TST     R1                      ; CHECK FOR ERROR
001501                          BNE     SNDAGN                  ; IF ERROR, TRY AGAIN
001501  050000  001473          ^O50000,SNDAGN
001503  104207  001520          MOV     #OUT.RQ,R0              ; WAIT FOR RESPONSE FROM HOST
001505  104201  000043          MOV     #BUFSIZ,R1
001507  060017                  XFC     MWR
001510  104200  177777  001520  MOV     #-1,OUT.RQ              ; MAKE REQUEST ILLEGAL
001513                          POP     <R2,R1,R0>
001513  104262                                                                  MOV (SP)+,R2
001514  104261                                                                  MOV (SP)+,R1
001515  104267                                                                  MOV (SP)+,R0
001516                          RETURN
001516  000000  000000          ^O00,0
```

```
                                ;STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS
                                ;
                                ;OUT BUFFER - DATA TO SEND TO HOST
                                ;
001520  000000  OUT.RQ: .WORD 0                ;HOST REQUEST CODE
001521  000000  OUT.01: .WORD 0                ;DATA ARGUMENT  1
001522  000000  OUT.02: .WORD 0                ;DATA ARGUMENT  2
001523  000000  OUT.03: .WORD 0                ;DATA ARGUMENT  3
001524  000000  OUT.04: .WORD 0                ;DATA ARGUMENT  4
001525  000000  OUT.05: .WORD 0                ;DATA ARGUMENT  5
001526  000000  OUT.06: .WORD 0                ;DATA ARGUMENT  6
001527  000000  OUT.07: .WORD 0                ;DATA ARGUMENT  7
001530  000000  OUT.08: .WORD 0                ;DATA ARGUMENT  8
001531  000000  OUT.09: .WORD 0                ;DATA ARGUMENT  9
001532  000000  OUT.10: .WORD 0                ;DATA ARGUMENT 10
001533  000000  OUT.11: .WORD 0                ;DATA ARGUMENT 11
001534  000000  OUT.12: .WORD 0                ;DATA ARGUMENT 12
001535  000000  OUT.13: .WORD 0                ;DATA ARGUMENT 13
001536  000000  OUT.14: .WORD 0                ;DATA ARGUMENT 14
001537  000000  OUT.15: .WORD 0                ;DATA ARGUMENT 15
001540  000000  OUT.16: .WORD 0                ;DATA ARGUMENT 16
001541  000000  OUT.17: .WORD 0                ;DATA ARGUMENT 17
001542  000000  OUT.18: .WORD 0                ;DATA ARGUMENT 18
001543  000000  OUT.19: .WORD 0                ;DATA ARGUMENT 19
001544  000000  OUT.20: .WORD 0                ;DATA ARGUMENT 20
001545  000000  OUT.21: .WORD 0                ;DATA ARGUMENT 21
001546  000000  OUT.22: .WORD 0                ;DATA ARGUMENT 22
001547  000000  OUT.23: .WORD 0                ;DATA ARGUMENT 23
001550  000000  OUT.24: .WORD 0                ;DATA ARGUMENT 24
001551  000000  OUT.25: .WORD 0                ;DATA ARGUMENT 25
001552  000000  OUT.26: .WORD 0                ;DATA ARGUMENT 26
001553  000000  OUT.27: .WORD 0                ;DATA ARGUMENT 27
001554  000000  OUT.28: .WORD 0                ;DATA ARGUMENT 28
001555  000000  OUT.29: .WORD 0                ;DATA ARGUMENT 29
001556  000000  OUT.30: .WORD 0                ;DATA ARGUMENT 30
001557  000000  OUT.31: .WORD 0                ;DATA ARGUMENT 31
001560  000000  OUT.32: .WORD 0                ;DATA ARGUMENT 32
001561  000000  OUT.33: .WORD 0                ;DATA ARGUMENT 33
001562  000000  OUT.34: .WORD 0                ;DATA ARGUMENT 34
        000043  BUFSIZ  =       . - OUT.RQ    ;SIZE OF BUFFER
```

```
                                        .SBTTL  TALK   - SDI LEVEL 2 INTERCHANGE ROUTINE
001563                          TALK:
                                ;
                                ;       TALK SENDS THE COMMAND TO THE DRIVE.  IF AN ERROR OCCURRS, R2 IS
                                ;       RETURNED NONZERO
                                ;
001563                                  PUSH    <R4,R5>         ; SAVE POINTER TO UNIT AND SUBUNIT PARAMETERS
001563  100464                                                                          MOV R4,-(SP)
001564  100465                                                                          MOV R5,-(SP)
001565  104302  003356                 MOV     SDI,R2          ; GET UNIT SDI SELECT MASK
001567  104205  000001                 MOV     #1,R5           ; SEND TIMEOUT DEFAULT IS ONE
001571  104207  003376                 MOV     #SNDONE,R0      ; MOVE SNDONE ADDRESS TO R0
001573  106037                         CMP     R3,R0           ; SEE IF ONLY TO BE SENT ONCE
001574                                  BCC     10$             ; IF SO, BRANCH
001574  040000  001600                 ^O040000,10$
001576  104205  001750                 MOV     #MAXSND,R5      ; SEND MAXIMUM NUMBER OF TIMES (ONLINE)
001600  104137                 10$:    MOV     (R3),R0         ; POINTS TO SDI COMMAND BUFFER
001601  104631  000001                 MOV     1(R3),R1        ; LOAD BYTE COUNT
001603  060004                         XFC     SEND            ; SEND SDI COMMAND
001604  115001                         TST     R1              ; SEE IF SDI COMMAND SENT SUCESSFULLY
001605                                  BEQ     15$             ; IF SO, BRANCH
001605  010000  001636                 ^O010000,15$
001607  117405                         DEC     R5              ; DECREMENT TIMEOUT
001610                                  BNE     10$             ; IF UNEXPIRED, BRANCH
001610  050000  001600                 ^O050000,10$
001612                                  POP     R5              ; RESTORE R5
001612  104265                                                                          MOV (SP)+,R5
001613                                  PUSH    R3              ; SAVE SDI PACKET POINTER
001613  100463                                                                          MOV R3,-(SP)
001614                                  HARDER  1
001614  104200  000000  001524                                                 MOV     #MS1,OUT.04
001617  104300  002212  001523                                                 MOV     LUNIT,OUT.03
001622  104202  105671                                                         MOV     #1!ERHARD+3000.,R2
001624  104020  001522                                                         MOV     R2,OUT.02
001626  104200  001626  001521                                                 MOV     #.,OUT.01
001631  104200  060013  001520                                                 MOV     #ERRMES,OUT.RQ
001634                                  BR      50$             ; BRANCH
001634  000000  002043                 ^O00,50$
001636                          15$:    POP     R5              ; RESTORE R5
001636  104265                                                                          MOV (SP)+,R5
001637  106203  003427                 CMP     #LONG,R3        ; SEE IF LONG TIMEOUT TO BE USED
001641                                  BMI     20$             ; IF SO, BRANCH
001641  070000  001647                 ^O070000,20$
001643  104304  002173                 MOV     SDISTO,R4       ; R4 HAS SHORT TIMEOUT
001645                                  BR      25$             ; BRANCH
001645  000000  001651                 ^O00,25$
001647  104304  002174          20$:    MOV     SDILTO,R4       ; R4 HAS LONG TIMEOUT
001651  104637  000002          25$:    MOV     2(R3),R0        ; POINT TO RECEIVE BUFFER
001653  104631  000003                 MOV     3(R3),R1        ; NUMBER OF WORDS IN RESPONSE
001655                                  PUSH    R3              ; SAVE POINTER TO COMMAND
001655  100463                                                                          MOV R3,-(SP)
001656  060005                         XFC     RCV             ; RECEIVE SDI RESPONSE
001657                                  POP     R3              ; RESTORE R3
001657  104263                                                                          MOV (SP)+,R3
001660  115001                         TST     R1              ; SEE IF SDI RESPONSE RECEIVED SUCESSFULLY
001661                                  BEQ     60$             ; IF SO, BRANCH
001661  010000  002122                 ^O010000,60$
```

```
     001663  106201  000001                  CMP     #1,R1           ; SEE IF TIMEOUT
     001665                                  BNE     30$             ; IF NOT, BRANCH
     001665  050000  001715                  ^O50000,30$
     001667  117404                          DEC     R4              ; DECREMENT TIMEOUT VALUE
     001670                                  BNE     25$             ; IF TIMEOUT UNEXPIRED, BRANCH
     001670  050000  001651                  ^O50000,25$
     001672                                  PUSH    R3              ; SAVE R3
     001672  100463                                                                  MOV R3,-(SP)
     001673                                  HARDER  2
     001673  104200  000014  001524                                          MOV     #MS2,OUT.04
     001676  104300  002212  001523                                          MOV     LUNIT,OUT.03
     001701  104202  105672                                                  MOV     #2!ERHARD+3000.,R2
     001703  104020  001522                                                  MOV     R2,OUT.02
     001705  104200  001705  001521                                          MOV     #.,OUT.01
     001710  104200  060013  001520                                          MOV     #ERRMES,OUT.RQ
     001713                                  BR      50$             ; BRANCH
     001713  000000  002043                  ^00,50$
     001715  106201  000002          30$:    CMP     #2,R1           ; SEE IF FIRST WORD NOT START FRAME
     001717                                  BNE     35$             ; IF NOT, BRANCH
     001717  050000  001743                  ^O50000,35$
     001721                                  HARDER  3
     001721  104200  000032  001524                                          MOV     #MS3,OUT.04
     001724  104300  002212  001523                                          MOV     LUNIT,OUT.03
     001727  104202  105673                                                  MOV     #3!ERHARD+3000.,R2
     001731  104020  001522                                                  MOV     R2,OUT.02
     001733  104200  001733  001521                                          MOV     #.,OUT.01
     001736  104200  060013  001520                                          MOV     #ERRMES,OUT.RQ
     001741                                  BR      50$             ; BRANCH
     001741  000000  002043                  ^00,50$
     001743  106201  000004          35$:    CMP     #4,R1           ; SEE IF FRAMING ERROR
     001745                                  BNE     40$             ; IF NOT, BRANCH
     001745  050000  001771                  ^O50000,40$
     001747                                  HARDER  4
     001747  104200  000063  001524                                          MOV     #MS4,OUT.04
     001752  104300  002212  001523                                          MOV     LUNIT,OUT.03
     001755  104202  105674                                                  MOV     #4!ERHARD+3000.,R2
     001757  104020  001522                                                  MOV     R2,OUT.02
     001761  104200  001761  001521                                          MOV     #.,OUT.01
     001764  104200  060013  001520                                          MOV     #ERRMES,OUT.RQ
     001767                                  BR      50$             ; BRANCH
     001767  000000  002043                  ^00,50$
     001771  106201  000010          40$:    CMP     #10,R1          ; SEE IF CHECKSUM ERROR
     001773                                  BNE     45$             ; IF NOT, BRANCH
     001773  050000  002017                  ^O50000,45$
     001775                                  HARDER  5
     001775  104200  000110  001524                                          MOV     #MS5,OUT.04
     002000  104300  002212  001523                                          MOV     LUNIT,OUT.03
     002003  104202  105675                                                  MOV     #5!ERHARD+3000.,R2
     002005  104020  001522                                                  MOV     R2,OUT.02
     002007  104200  002007  001521                                          MOV     #.,OUT.01
     002012  104200  060013  001520                                          MOV     #ERRMES,OUT.RQ
     002015                                  BR      50$             ; BRANCH
     002015  000000  002043                  ^00,50$
     002017  106201  000020          45$:    CMP     #20,R1          ; SEE IF BUFFER TOO SMALL
     002021                                  BNE     55$             ; IF NOT, BRANCH
     002021  050000  002062                  ^O50000,55$
     002023                                  HARDER  6
```

```
                002023  104200  000135  001524                                          MOV     #MS6,OUT.04
                002026  104300  002212  001523                                          MOV     LUNIT,OUT.03
                002031  104202  105676                                                  MOV     #6!ERHARD+3000.,R2
                002033  104020  001522                                                  MOV     R2,OUT.02
                002035  104200  002035  001521                                          MOV     #.,OUT.01
                002040  104200  060013  001520                                          MOV     #ERRMES,OUT.RQ
     60         002043                  50$:    CERROR  5,<#SER00,COMND>
                002043  104200  002573  001525                                          MOV     #SER00,OUT.05
                002046  104300  003357  001526                                          MOV     COMND,OUT.06
     61         002051                          ENDERR  7               ; FLAG END OF REPORTING BUFFER
                002051  104200  003276  001527                                          MOV     #SER22,OUT.07
                002054  104200  000010  003626                                          MOV     #7+1,ERRPOS     ; SET THE POSITION
     62         002057                          POP     R3              ; SAVE SDI PACKET POINTER
                002057  104263                                                                  MOV (SP)+,R3
     63         002060                          BR      65$             ; EXIT
                002060  000000  002170                  ^O00,65$
     64         002062                  55$:    HARDER  7               ; UNKNOWN ERROR RETURNED BY UDA
                002062  104200  000160  001524                                          MOV     #MS7,OUT.04
                002065  104300  002212  001523                                          MOV     LUNIT,OUT.03
                002070  104202  105677                                                  MOV     #7!ERHARD+3000.,R2
                002072  104020  001522                                                  MOV     R2,OUT.02
                002074  104200  002074  001521                                          MOV     #.,OUT.01
                002077  104200  060013  001520                                          MOV     #ERRMES,OUT.RQ
     65         002102                          CERROR  5,<R1,#SER00,COMND>     ; REPORT ERROR
                002102  104010  001525                                          MOV     R1,OUT.05
                002104  104200  002573  001526                                          MOV     #SER00,OUT.06
                002107  104300  003357  001527                                          MOV     COMND,OUT.07
     66         002112                          ENDERR  8               ; FLAG END OF REPORTING BUFFER
                002112  104200  003276  001530                                          MOV     #SER22,OUT.08
                002115  104200  000011  003626                                          MOV     #8+1,ERRPOS     ; SET THE POSITION
     67         002120                          BR      65$             ; EXIT
                002120  000000  002170                  ^O00,65$
     68         002122  114002                  60$:    CLR     R2              ; FLAG AS NO ERROR OCCURED
     69         002123  106637  000004                  CMP     4(R3),R0        ; SEE IF COMMAND ACCEPTED
     70         002125                          BEQ     65$             ; IF SO, BRANCH
                002125  010000  002170                  ^O10000,65$
     71         002127                          HARDER  8
                002127  104200  000221  001524                                          MOV     #MS8,OUT.04
                002132  104300  002212  001523                                          MOV     LUNIT,OUT.03
                002135  104202  105700                                                  MOV     #8!ERHARD+3000.,R2
                002137  104020  001522                                                  MOV     R2,OUT.02
                002141  104200  002141  001521                                          MOV     #.,OUT.01
                002144  104200  060013  001520                                          MOV     #ERRMES,OUT.RQ
     72         002147                          CERROR  5,<#SER00,COMND,4(R3),R0> ; REPORT FURTHER ERRORS
                002147  104200  002573  001525                                          MOV     #SER00,OUT.05
                002152  104300  003357  001526                                          MOV     COMND,OUT.06
                002155  104630  000004  001527                                          MOV     4(R3),OUT.07
                002160  104070  001530                                                  MOV     R0,OUT.08
     73         002162                          ENDERR  9
                002162  104200  003276  001531                                          MOV     #SER22,OUT.09
                002165  104200  000012  003626                                          MOV     #9+1,ERRPOS     ; SET THE POSITION
     74         002170                  65$:    POP     R4              ; RESTORE R4
                002170  104264                                                                  MOV (SP)+,R4
     75         002171                          RETURN
                002171  000000  000000                  ^O00,0
     76
     77         002173  000012          SDISTO: .WORD   10.             ; SDI SHORT TIMEOUT
```

```
     78 002174  000024                SDILTO: .WORD   20.          ; SDI LONG TIMEOUT
```

```
                                                .SBTTL  TO - CALCULATE TIMEOUT INTERVALS
002175                                  TO:
                                        ;
                                        ;       CALCULATE THE TIMEOUT IN 9SEC INTERVALS (SDI RECEIVE XFC TAKES
                                        ;       9 SEC)
                                        ;
002175  104201  000001                          MOV     #1,R1           ; SET UP LOG2 SHIFTER
002177  105011                         1$:     ADD     R1,R1           ; DOUBLE THE TIMEOUT VALUE
002200  117407                                 DEC     R0              ; DECREMENT COUNT
002201                                          BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
002201  050000  002177                          ^O50000,1$
002203  115407                         2$:     INC     R0              ; INCREMENT 9 SEC COUNT
002204  107201  000011                          SUB     #9.,R1          ; SUBTRACT 9 SEC FROM TIMEOUT
002206                                          BPL     2$              ; IF MORE TIME TO GO, BRANCH
002206  030000  002203                          ^O30000,2$
002210                                          RETURN                  ; RETURN TO CALLING PROGRAM
002210  000000  000000                          ^O0,0

002212  177777                          LUNIT:  .WORD -1                        ;LOGICAL UNIT NUMBER (-1 FOR NOT AVAILABLE)

                                                .SBTTL  LINIT - INIT THE DRIVE
002213  104302  003356                  LINIT:  MOV     SDI,R2          ; R2 HAS INTERCONNECT CODE
002215  060011                                  XFC     DINIT
002216                                          RETURN
002216  000000  000000                          ^O0,0
```

```
                                        .SBTTL  STACK AREA
                                ;STACK AREA
                                
002220  123456                          .WORD 123456            ;END MARKER FOR STACK
002221                                  .BLKW 31.               ;STACK
002260  123456                  STACK:  .WORD 123456            ;MARKER FOR STACK UNDERFLOW
                                
        003516                  SUB     =       SUB + CR        ;MODIFY SUB TO POINT AT SUBUNIT CHAR
        005670                          ERRN=3000.              ;START ERROR NUMBERS AT 3000.
                                
002261  003274                  SER18E: .WORD   SER18A          ; FOR MULTIPLE SUBUNIT ERROR REPORTING
002262  003271                          .WORD   SER18B
002263  003266                          .WORD   SER18C
002264  003263                          .WORD   SER18D
                                
```

```
                                        .SBTTL  TEST 3 START AND LOOP ON UNITS
                                ;SEQUENCE THE DIAGNOSTICS TO ALL UNITS SELECTED.
                                ;TEST CODE WILL BE CALLED FOR EACH DISK SUBUNIT TO BE TESTED.
                                ;  LUNIT WILL CONTAIN LOGICAL UNIT NUMBER OF DRIVE FOR ERROR REPORTS
                                ;  SDI WILL CONTAIN SDI INTERCONNECT CODE FOR SELECTED DRIVE
                                ;  UNITNB WILL CONTAIN AN EVEN NUMBER FOR TESTING FIRST SUBUNIT OF A DRIVE
                                ;                      AN ODD NUMBER FOR TESTING SECOND SUBUNIT OF A DRIVE
                                
 002265  114001                 T3STRT: CLR     R1              ;START WITH UNIT 0 INDEX
                                
 002266  104207  003335         PORT2:  MOV     #UNITS,R0       ; GET POINTER TO UNITS TABLE
 002270  105017                         ADD     R1,R0           ; ADD INDEX
 002271  104173                         MOV     (R0),R3         ; GET CONTENTS OF TABLE
 002272                                 BPL     1$              ; IF THIS UNIT IS PRESENT, BRANCH
         002272  030000  002304         ^O030000,1$
 002274  102201  000003                 BIT     #3,R1           ; SEE IF ON SUBUNIT 0 OF UNIT
 002276                                 BNE     PORT5           ; IF NOT, TEST NEXT SUBUNIT
         002276  050000  002342         ^O050000,PORT5
 002300  105201  000003                 ADD     #3,R1           ; IF NO SUBUNIT 0, THEN NO UNIT - SKIP OTHER SUBUNITS
 002302                                 BR      PORT5           ; BYPASS IF NO UNIT
         002302  000000  002342         ^O00,PORT5
 002304  102203  040000         1$:     BIT     #40000,R3       ; SEE IF THIS UNIT IS TO BE TESTED
 002306                                 BNE     PORT5           ; IF NOT, BRANCH
         002306  050000  002342         ^O050000,PORT5
 002310  104010  003355                 MOV     R1,UNITNB       ; STORE UNIT INDEX
 002312  104030  002212                 MOV     R3,LUNIT        ; STORE LOGICAL UNIT NUMBER FOR DRIVE
 002314  104202  000001                 MOV     #UNIT0,R2       ; GET UNIT 0 INTERCONNECT CODE
 002316  110601                         ROR     R1              ; DIVIDE UNITNB BY FOUR
 002317  110601                         ROR     R1
 002320  103201  177760                 BIC     #LBLONB,R1      ; CLEAR UNUSED BITS
 002322  117401                 PORT3:  DEC     R1
 002323                                 BMI     PORT4           ; FOR EACH DRIVE OVER 0 SHIFT R2 LEFT
         002323  070000  002332         ^O070000,PORT4
 002325  110202                         ROL     R2
 002326  103202  000001                 BIC     #1,R2           ; CLEAR CARRY ROTATED INTO REG (IF ANY)
 002330                                 BR      PORT3
         002330  000000  002322         ^O00,PORT3
 002332  104020  003356         PORT4:  MOV     R2,SDI          ; STORE SDI INTERCONNECT CODE
 002334                                 BR      STRTST
         002334  000000  005774         ^O00,STRTST
 002336  104206  002260         TESTX:  MOV     #STACK,SP       ; RESET STACK DUE TO JUMPS OUT OF
                                                                ; SUBROUTINES
 002340  104301  003355                 MOV     UNITNB,R1       ; GET UNIT INDEX
 002342  115401                 PORT5:  INC     R1              ; INCREMENT INDEX
 002343  106201  000020                 CMP     #16.,R1         ; CHECK IF 16 DRIVES ALREADY SELECTED
 002345                                 BNE     PORT2           ; REPEAT FOR ALL DRIVES
         002345  050000  002266         ^O050000,PORT2
 002347  104207  060016         DONECD: MOV     #DONE,R0        ; END OF PROGRAM
 002351                                 CALL    HOSTRQ
         002351  020000  001463         ^O020000,HOSTRQ
 002353                                 BR      DONECD          ; REPEAT IF RETURNED
         002353  000000  002347         ^O00,DONECD
```

ERROR EXIT

```
      1                                                .SBTTL  ERROR EXIT
      2 002355                                 TESTEW: CERROR  5,<#SER23,R1>          ; PRINT REAL TIME DRIVE STATE ONLY
        002355  104200  003344  001525                         MOV     #SER23,OUT.05
        002360  104010  001526                                 MOV     R1,OUT.06
      3 002362                                         BR      TESTED
        002362  000000  002400                         ^O0,TESTED
      4 002364                                 TESTEV: ENDERR  5
        002364  104200  003276  001525                         MOV     #SER22,OUT.05
        002367  104200  000006  003626                         MOV     #5+1,ERRPOS     ; SET THE POSITION
      5 002372  104203  001520                 TESTEX: MOV     #OUT.RQ,R3             ; SET UP POSITION IN R3
      6 002374  105303  003626                         ADD     ERRPOS,R3
      7 002376                                 TESTEY: CALL    STRST                  ; GET STATUS FROM ST
        002376  020000  002572                         ^O020000,STRST
      8 002400  104302  003356                 TESTED: MOV     SDI,R2                 ; R2 HAS INTERCONNECT CODE
      9 002402  060011                                 XFC     DINIT                  ; AND INIT DRIVE
     10 002403  104307  001520                         MOV     OUT.RQ,R0              ; PRINT ERROR
     11 002405                                         CALL    HOSTRQ
        002405  020000  001463                         ^O020000,HOSTRQ
     12 002407  104200  000377  003451   DR.CLR: MOV     #LOBYTE,ERRORS         ; CLEAR ALL ERRORS
     13 002412  104203  003372                         MOV     #CR.CLR,R3
     14 002414                                         CALL    TALK
        002414  020000  001563                         ^O020000,TALK
     15 002416                                         RETURN                         ; GO TO CALLING ROUTINE
        002416  000000  000000                         ^O0,0
```

```
                                        .SBTTL  FNDCYL - FIND CYLINDER
                                ;
                                ;       FNDCYL  FIND CYLINDER
                                ;
                                ;       INPUTS: R4 -> HIGHEST CYL (VAR1)
                                ;               R5 -> DESIRED BN (VAR2)
                                ;               R3 = VAR3
                                ;               R1 = VAR4
                                ;
002420                          FNDCYL:
002420  106200  000104  003625          CMP     #'D,LETTER              ;IS IT A DIAGNOSTIC BLOCK?
002423                                  BNE     FND3
002423  050000  002440                  ^O50000,FND3
                                ; *** FOR DBN AREA
002425  104205  003572                  MOV     #LCDBN,R5
002427  104204  003577          FNDCY2: MOV     #FDIACYL,R4
002431  103200  170000  003600          BIC     #^CHBHINB,FDIACYL+1
002434  104303  003603                  MOV     SECTRK,R3
002436                                  BR      FND4
002436  000000  002470                  ^O0,FND4
002440  106200  000114  003625  FND3:   CMP     #'L,LETTER              ;IS IT A LOGICAL BLOCK?
002443                                  BNE     1$                      ;IF NOT, BRANCH
002443  050000  002461                  ^O50000,1$
                                ; *** FOR LBN AREA
002445  114000  003606                  CLR     TSTCYL
002447  114000  003607                  CLR     TSTCYL+1
002451  104204  003606                  MOV     #TSTCYL,R4
002453  104303  003527                  MOV     SUB+LBNTRK,R3           ; R3 = LBN'S PER TRACK (VAR3)
002455  103203  177400                  BIC     #HIBYTE,R3
002457                                  BR      FND4                    ;CONTINUE
002457  000000  002470                  ^O0,FND4
                                ; *** FOR XBN AREA
002461  104204  003601          1$:     MOV     #FXBNCYL,R4             ;SET POINTER TO FIRST XBN CYL->VAR2
002463  103200  170000  003602          BIC     #^CHBHINB,FXBNCYL+1
002466  104303  003603                  MOV     SECTRK,R3               ;R3 = SECTORS PER TRACK (VAR3)
002470  104302  003517          FND4:   MOV     SUB+LBNCYL+1,R2
002472  103202  007777                  BIC     #HBHINB,R2
002474  101642  000001                  BIS     1(R4),R2
002476  100642  000001                  MOV     R2,1(R4)
002500                          4$:
002500  104247                          MOV     (R4)+,R0                ;START LOADING VARIABLES
002501  104070  003633                  MOV     R0,VAR1
002503  104147                          MOV     (R4),R0
002504  104070  003634                  MOV     R0,VAR1+1
002506  104257                          MOV     (R5)+,R0
002507  104070  003635                  MOV     R0,VAR2
002511  104157                          MOV     (R5),R0
002512  104070  003636                  MOV     R0,VAR2+1
002514  104030  003637                  MOV     R3,VAR3
002516  104010  003640                  MOV     R1,VAR4
002520  104207  003633                  MOV     #VAR1,R0                ; R0 -> VARIABLES
002522  104201  003516                  MOV     #SUB,R1                 ; R1 -> SUBUNIT CHARACTERISTICS
002524  060020                          XFC     CVT                     ;CONVERT TO CYLINDER VALUE
002525  106200  000104  003625          CMP     #'D,LETTER              ; DON'T COMPARE IF IN DBN AREA
002530                                  BEQ     7$
002530  010000  002557                  ^O10000,7$
002532  106300  003643  003632          CMP     GROUP,OLDGRP            ; HAVE WE GONE OVER 2 CYL BOUNDARIES?
```

FNDCYL - FIND CYLINDER

```
     53 002535                                          BPL     6$                      ; IF NOT, BRANCH
        002535  030000  002554                         ^030000,6$
     54 002537  107300  003613  003604                 SUB     BLOCKC,TSTBLK           ; ELSE, GO BACK ONE CYLYINDER
     55 002542                                          BCC     5$                      ; IF NOT CARRY, BRANCH
        002542  040000  002546                         ^040000,5$
     56 002544  117400  003605                         DEC     TSTBLK+1
     57 002546  114000  003632           5$:           CLR     OLDGRP                  ; NEW GROUP
     58 002550  117404                                 DEC     R4                      ; RESET POINTERS
     59 002551  117405                                 DEC     R5
     60 002552                                          BR      FNDCYL                  ; AND TRY AGAIN
        002552  000000  002420                         ^00,FNDCYL
     61
     62 002554  104300  003643  003632  6$:           MOV     GROUP,OLDGRP            ; SAVE IN OLD GROUP FOR NEXT TIME THROUGH
     63 002557  104300  003641  003606  7$:           MOV     CYLLO,TSTCYL            ;STORE IN TSTCYL
     64 002562  104300  003642  003607                 MOV     CYLLO+1,TSTCYL+1
     65 002565  104300  003643  003610                 MOV     GROUP,TSTCYL+2
     66 002570                                          RETURN
        002570  000000  000000                         ^00,0
```

STRST - STORE INFORMATION IN AREA 'ST'

```
                                        .SBTTL  STRST - STORE INFORMATION IN AREA 'ST'
                                ;
                                ;       STRST   STORE ST
                                ;
                                ;       INPUT   R3 -> OUTPUT BUFFER
                                ;       OUTPUT  OUTPUT BUFFER FILLED WITH VALUES FROM ST
                                ;
 002572                         STRST:  PUSH    <R0,R1,R2>
 002572  100467                                                                  MOV R0,-(SP)
 002573  100461                                                                  MOV R1,-(SP)
 002574  100462                                                                  MOV R2,-(SP)
 002575  104302  003356                  MOV     SDI,R2                  ;R2 HAS INTERCONNECT CODE
 002577  060007                          XFC     STATUS                  ;GET DRIVE STATE
 002600  103201  077674                  BIC     #077674,R1              ;CLEAR UNUSED BITS
 002602  100231                          MOV     R1,(R3)+                ; STORE IN OUT BUFFER
 002603  104202  000007                  MOV     #7,R2                   ; R2 IS # OF WORDS TO MOVE
 002605  104207  003503                  MOV     #ST+7,R0                ; R0 -> STATUS
 002607  104471                  1$:     MOV     -(R0),R1
 002610  100231                          MOV     R1,(R3)+
 002611  117402                          DEC     R2
 002612                                  BNE     1$
 002612  050000  002607                  ^O50000,1$
 002614                                  POP     <R2,R1,R0>
 002614  104262                                                                  MOV (SP)+,R2
 002615  104261                                                                  MOV (SP)+,R1
 002616  104267                                                                  MOV (SP)+,R0
 002617                                  RETURN
 002617  000000  000000                  ^O0,0
```

```
                                        .SBTTL  SEEK
                                ;SEEK
                                ;
                                ;SEEK TO CYLINDER POINTED TO BY CONTENTS OF R1
                                ;
                                ;INPUTS:
                                ;       R1 - POINTER TO CYLINDER NUMBER
                                ;       R2 - SDI INTERCONNECT
                                ;
002621                          SEEK:   PUSH <R0,R1,R3>
002621  100467                                                                  MOV R0,-(SP)
002622  100461                                                                  MOV R1,-(SP)
002623  100463                                                                  MOV R3,-(SP)
002624  104302  003356                 MOV     SDI,R2
002626  104210  003461                 MOV (R1)+,INS+1                 ;PUT CYLINDER INTO COMMAND
002630  104210  003462                 MOV (R1)+,INS+2
002632  104110  003463                 MOV (R1),INS+3                  ;PUT GROUP INTO COMMAND
002634  104203  003423         SEEKA:  MOV #CR.SEK,R3                  ;POINT TO COMMAND
002636  104200  002706  003357         MOV     #MS.SEK,COMND           ; SET UP FOR ERROR
002641                          ;       CALL    TALK                    ; INITIATE SDI INTERCHANGE
002641  020000  001563                 ^O20000,TALK
002643  115002                  ;       TST     R2                      ; SEE IF ERROR OCCURRED
002644                          ;       BEQ     SEEK1                   ; IF NOT, BRANCH
002644  010000  002652                 ^O10000,SEEK1
002646                          ;       CALL    TESTEX                  ; IF SO, REPORT
002646  020000  002372                 ^O20000,TESTEX
002650                          ;       BR      SEEK3                   ; AND EXIT
002650  000000  003006                 ^O0,SEEK3
002652  114003                  SEEK1:  CLR R3                          ;SET UP WORST CASE SEEK TIME
002653  104302  003356          SEEK2:  MOV     SDI,R2                  ; MOVE MASK TO R2
002655                          ;       CALL    RDSTAT                  ; GET DRIVE STATUS
002655  020000  001432                 ^O20000,RDSTAT
002657  115002                  ;       TST     R2                      ; WAS IT OK?
002660                          ;       BEQ     2$                      ; IF SO, BRANCH
002660  010000  002736                 ^O10000,2$
002662  102201  000400                 BIT     #RCVERR,R1              ; SEE WHICH ERROR
002664                          ;       BNE     1$                      ; AND BRANCH
002664  050000  002712                 ^O50000,1$
002666                          ;       HARDER  10                      ; REPORT INVALID STATUS ERROR
002666  104200  000341  001524                 MOV     #MS10,OUT.04
002671  104300  002212  001523                 MOV     LUNIT,OUT.03
002674  104202  105702                         MOV     #10!ERHARD+3000.,R2
002676  104020  001522                         MOV     R2,OUT.02
002700  104200  002700  001521                 MOV     #.,OUT.01
002703  104200  060013  001520                 MOV     #ERRMES,OUT.RQ
002706                          ;       CALL    TESTEW                  ; BRANCH TO DONE
002706  020000  002355                 ^O20000,TESTEW
002710                          ;       BR      SEEK3
002710  000000  003006                 ^O0,SEEK3
002712                          1$:
002712                          ;       HARDER  12                      ; REPORT XMIT ERROR
002712  104200  000434  001524                 MOV     #MS12,OUT.04
002715  104300  002212  001523                 MOV     LUNIT,OUT.03
002720  104202  105704                         MOV     #12!ERHARD+3000.,R2
002722  104020  001522                         MOV     R2,OUT.02
002724  104200  002724  001521                 MOV     #.,OUT.01
002727  104200  060013  001520                 MOV     #ERRMES,OUT.RQ
```

```
34 002732
   002732  020000  002355                    CALL    TESTEW                  ; BRANCH TO DONE
                                              ^020000,TESTEW
35 002734
   002734  000000  003006                    BR      SEEK3
                                              ^00,SEEK3
36 002736                              2$:
37 002736  102201  100000                    BIT #RWRDY,R1                   ;IS R/W READY SET?
38 002740                                     BNE SEEK3                       ;  YES
   002740  050000  003006                    ^050000,SEEK3
39 002742  115403                           INC R3                          ;BUMP COUNT
40 002743                                    BNE SEEK2                       ;KEEP WAITING
   002743  050000  002653                    ^050000,SEEK2
41 002745                                    HARDER  27,<INS+1,INS+2,INS+3>  ;SEEK COMPLETE TIME OUT
   002745  104200  001633  001524                      MOV     #MS27,OUT.04
   002750  104300  003461  001525                      MOV     INS+1,OUT.05
   002753  104300  003462  001526                      MOV     INS+2,OUT.06
   002756  104300  003463  001527                      MOV     INS+3,OUT.07
   002761  104300  002212  001523                      MOV     LUNIT,OUT.03
   002764  104202  105723                            MOV     #27!ERHARD+3000.,R2
   002766  104020  001522                            MOV     R2,OUT.02
   002770  104200  002770  001521                    MOV     #.,OUT.01
   002773  104200  060013  001520                    MOV     #ERRMES,OUT.RQ
42
43                                   ;       MOV     OUT.RQ,R0
                                     ;       CALL    HOSTRQ
44 002776                                    ENDERR  8
   002776  104200  003276  001530                      MOV     #SER22,OUT.08
   003001  104200  000011  003626                      MOV     #8+1,ERRPOS     ; SET THE POSITION
45 003004                                    CALL    TESTEX
   003004  020000  002372                    ^020000,TESTEX
46 003006                              SEEK3: POP <R3,R1,R0>
   003006  104263                                                      MOV (SP)+,R3
   003007  104261                                                      MOV (SP)+,R1
   003010  104267                                                      MOV (SP)+,R0
47 003011                                    RETURN
   003011  000000  000000                    ^00,0
```

```
                                        .SBTTL  READ1 - READ SECTORS ON A TRACK FOR TEST
                                ;READ1
                                ;
                                ;READ SECTORS ON THE SELECTED TRACK UNTIL ONE IS READ CORRECTLY
                                ;OR THE MAXIMUM NUMBER OF SECTORS IS READ
                                ;
                                ;INPUTS:
                                ;       R3 HAS MAX NUMBER OF SECTORS TO READ
                                ;       TRACK HAS TRACK NUMBER
                                ;       R5 -> POINTER TO BLOCK NUMBER OF FIRST SECTOR
                                ;OUTPUTS:
                                ;       R2 IS CLOBBERED
                                ;
003013                          READ1:  PUSH <R0,R1,R3,R4,R5>
003013  100467                                                                  MOV R0,-(SP)
003014  100461                                                                  MOV R1,-(SP)
003015  100463                                                                  MOV R3,-(SP)
003016  100464                                                                  MOV R4,-(SP)
003017  100465                                                                  MOV R5,-(SP)
003020  104304  003644                MOV     TRACK,R4                ;R4 IS TRACK
003022  104207  004373                MOV #RBUF0+RW.LOW,R0
003024  104251                        MOV (R5)+,R1                    ;PUT CLOCK NUMBER IN
003025  100271                        MOV R1,(R0)+                    ;  READ BUFFER
003026  104151                        MOV (R5),R1
                                
003027  106200  000130  003625        CMP     #'X,LETTER              ; DO WE DO PROCESS FOR XBN?
003032                                BNE     9$                      ; IF NOT, BRANCH
003032  050000  003050                ^050000,9$
                                ; *** FOR XBN AREA
003034  104305  003520                MOV     SUB+HIXBN,R5            ; DO SPECIAL PROCESSING TO PUT HEADERIN PLACE
003036  110605                        ROR     R5                      ; SHIFT TO NEXT NIBBLE
003037  110605                        ROR     R5
003040  110605                        ROR     R5
003041  110605                        ROR     R5
003042  103205  170377                BIC     #HBLONB,R5              ; STRIP OFF UNUSED PORTION
003044  101205  120000                BIS     #HD.XBN,R5              ; SET HEADER CODE
003046                                BR      10$
003046  000000  003075                ^00,10$
003050  106200  000104  003625  9$:   CMP     #'D,LETTER              ; DO WE PROCESS FOR DBN?
003053                                BEQ     11$                     ; IF SO, BRANCH
003053  010000  003063                ^010000,11$
                                ; *** FOR LBN AREA
003055  104305  003520                MOV     SUB+HILBN,R5            ; GET HI LBN VALUE (DON'T HAVE TO ROR)
003057  103205  170377                BIC     #HBLONB,R5
003061                                ASSUME HD.LBN,0
003061                                BR      10$
003061  000000  003075                ^00,10$
                                ; *** FOR DBN AREA
003063  104305  003521        11$:    MOV     SUB+HIDBN,R5            ; GET HIGH ORDER BITS OF STARTING DBN
003065  110605                        ROR     R5                      ; MOVE TO CORRECT POSITION
003066  110605                        ROR     R5
003067  110605                        ROR     R5
003070  110605                        ROR     R5
003071  103205  170377                BIC     #HBLONB,R5              ; CLEAR UNUSED BITS
003073  101205  140000                BIS     #HD.DBN,R5              ; SET HEADER CODE
003075  105051                10$:    ADD     R5,R1
003076  100271                        MOV R1,(R0)+
```

```
     49 003077  104171                          MOV (R0),R1             ;PUT IN TRACK NUMBER
     50 003100  103201  000377                   BIC #LOBYTE,R1          ;(MERGE WITH REAL-TIME COMMAND)
     51 003102  101041                          BIS R4,R1
     52 003103  100171                          MOV R1,(R0)
     53 003104  104207  004370          READ1A: MOV #RBUF0,R0           ;POINT TO READ BUFFER
     54 003106  104302  003356                  MOV     SDI,R2          ; MOVE MASK TO R2
     55 003110                                  CALL    RDSTAT          ; GET DRIVE STATUS
        003110  020000  001432                  ^O20000,RDSTAT
     56 003112  115002                          TST     R2              ; WAS IT OK?
     57 003113                                  BEQ     2$              ; IF SO, BRANCH
        003113  010000  003171                  ^O10000,2$
     58 003115  102201  000400                  BIT     #RCVERR,R1      ; SEE IF ANY ERRORS
     59 003117                                  BNE     1$              ; REPORT
        003117  050000  003145                  ^O50000,1$
     60 003121                                  HARDER  10              ; REPORT INVALID STATUS ERROR
        003121  104200  000341  001524                  MOV     #MS10,OUT.04
        003124  104300  002212  001523                  MOV     LUNIT,OUT.03
        003127  104202  105702                          MOV     #10!ERHARD+3000.,R2
        003131  104020  001522                          MOV     R2,OUT.02
        003133  104200  003133  001521                  MOV     #.,OUT.01
        003136  104200  060013  001520                  MOV     #ERRMES,OUT.RQ
     61 003141                                  CALL    TESTEW          ; BRANCH TO DONE
        003141  020000  002355                  ^O20000,TESTEW
     62 003143                                  BR      READ1X
        003143  000000  003326                  ^O0,READ1X
     63 003145                          1$:
     64 003145                                  HARDER  12              ; REPORT XMIT ERROR
        003145  104200  000434  001524                  MOV     #MS12,OUT.04
        003150  104300  002212  001523                  MOV     LUNIT,OUT.03
        003153  104202  105704                          MOV     #12!ERHARD+3000.,R2
        003155  104020  001522                          MOV     R2,OUT.02
        003157  104200  003157  001521                  MOV     #.,OUT.01
        003162  104200  060013  001520                  MOV     #ERRMES,OUT.RQ
     65 003165                                  CALL    TESTEW          ; BRANCH TO DONE
        003165  020000  002355                  ^O20000,TESTEW
     66 003167                                  BR      READ1X
        003167  000000  003326                  ^O0,READ1X
     67 003171                          2$:
     68 003171  102201  100000                  BIT #RWRDY,R1           ; SEE IF READ WRITE READY IS STILL HIGH
     69 003173                                  BNE READ1C              ; IF SO, BRANCH
        003173  050000  003221                  ^O50000,READ1C
     70 003175                                  HARDER  24              ;READ/WRITE READY DROPPED BEFORE READ
        003175  104200  001306  001524                  MOV     #MS24,OUT.04
        003200  104300  002212  001523                  MOV     LUNIT,OUT.03
        003203  104202  105720                          MOV     #24!ERHARD+3000.,R2
        003205  104020  001522                          MOV     R2,OUT.02
        003207  104200  003207  001521                  MOV     #.,OUT.01
        003212  104200  060013  001520                  MOV     #ERRMES,OUT.RQ
     71 003215                                  CALL    TESTEV          ; BRANCH
        003215  020000  002364                  ^O20000,TESTEV
     72 003217                                  BR      READ1X
        003217  000000  003326                  ^O0,READ1X
     73 003221  104302  003356          READ1C: MOV     SDI,R2
     74 003223  060012                          XFC WAITSI              ;WAIT FOR SECTOR OR INDEX PULSE B4 READ
     78 003224  104202  000400                  MOV     #BUFFSZ,R2      ; R2 = BUFFER SIZE FOR 52
     82 003226  060002                          XFC XREAD               ;READ THE SECTOR
     83 003227  115001                          TST R1                  ;CHECK FOR ERROR
```

```
 84 003230                                          BEQ READ1X                      ;END ROUTINE IF READ OK
    003230  010000  003326                          ^O10000,READ1X
 85 003232  117403                                  DEC R3                          ;COUNT MAX SECTORS TO READ
 86 003233                                          BEQ READ1E                      ;REPORT ERROR IF ALL READ
    003233  010000  003251                          ^O10000,READ1E
 87 003235  104207  004373                          MOV #RBUF0+RW.LOW,R0
 88 003237  104171                                  MOV (R0),R1                     ;INCREMENT BLOCK NUMBER
 89 003240  115401                                  INC R1
 90 003241  100271                                  MOV R1,(R0)+
 91 003242                                          BCC READ1B
    003242  040000  003247                          ^O40000,READ1B
 92 003244  104171                                  MOV (R0),R1
 93 003245  115401                                  INC R1
 94 003246  100171                                  MOV R1,(R0)
 95 003247                              READ1B: BR READ1A                           ;GO READ NEXT SECTOR
    003247  000000  003104                          ^O0,READ1A
 96 003251                              READ1E: HARDER  28,<LETTER,RBUF0+RW.LOW,RBUF0+RW.HI,INS+1,INS+2,INS+3,RBUF0+RW.CMD>
    003251  104200  001716  001524                                                  MOV     #MS28,OUT.04
    003254  104300  003625  001525                                                  MOV     LETTER,OUT.05
    003257  104300  004373  001526                                                  MOV     RBUF0+RW.LOW,OUT.06
    003262  104300  004374  001527                                                  MOV     RBUF0+RW.HI,OUT.07
    003265  104300  003461  001530                                                  MOV     INS+1,OUT.08
    003270  104300  003462  001531                                                  MOV     INS+2,OUT.09
    003273  104300  003463  001532                                                  MOV     INS+3,OUT.10
    003276  104300  004375  001533                                                  MOV     RBUF0+RW.CMD,OUT.11
    003301  104300  002212  001523                                                  MOV     LUNIT,OUT.03
    003304  104202  105724                                                          MOV     #28!ERHARD+3000.,R2
    003306  104020  001522                                                          MOV     R2,OUT.02
    003310  104200  003310  001521                                                  MOV     #.,OUT.01
    003313  104200  060013  001520                                                  MOV     #ERRMES,OUT.RQ
 97 003316                                          ENDERR  12
    003316  104200  003276  001534                                                  MOV     #SER22,OUT.12
    003321  104200  000015  003626                                                  MOV     #12+1,ERRPOS    ; SET THE POSITION
 98 003324                                          CALL    TESTEX
    003324  020000  002372                          ^O20000,TESTEX
 99                                     ;           MOV     OUT.RQ,R0
100                                     ;           CALL    HOSTRQ
101 003326                              READ1X: POP <R5,R4,R3,R1,R0>
    003326  104265                                                                      MOV (SP)+,R5
    003327  104264                                                                      MOV (SP)+,R4
    003330  104263                                                                      MOV (SP)+,R3
    003331  104261                                                                      MOV (SP)+,R1
    003332  104267                                                                      MOV (SP)+,R0
102 003333                                          RETURN
    003333  000000  000000                          ^O0,0
```

```
                                        .SBTTL  UNITS, SDI COMMANDS AND PROGRAM VARIABLES
                                ;PROGRAM VARIABLES

                                ;UNIT NUMBER STORAGE FOR DISK DRIVES TO TEST

003335  000020                UNITS:  .REPT   16.                     ; 16 SUBUNITS (8 MAX) 4 ON EACH UNIT
                                        .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
                                        .ENDR
003335  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003336  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003337  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003340  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003341  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003342  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003343  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003344  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003345  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003346  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003347  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003350  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003351  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003352  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003353  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
003354  177777                          .WORD   177777                  ; MARK ALL UNITS AS NON-EXISTANT TO START WITH

003355  000000                UNITNB: .WORD 0                         ;NUMBER OF UNIT CURRENTLY UNDER TEST
                                                                      ;POINTER TO TABLE ABOVE

003356  000000                SDI:    .WORD 0                         ;SDI INTERCONNECT CODE FOR XFC CALLS

003357  000000                COMND:  .WORD   0                       ; POINTER TO COMMAND MESSAGE
                                ;       MESSAGE TABLES

C03360                          CR.ONL: MSG     ONL,2,ST,7,COMPLT       ; DRIVE ONLINE
003360  003442                          .WORD ONL                       ;ADDRESS OF COMMAND
003361  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
003362  003474                          .WORD ST                        ;ADDRESS OF REPLY
003363  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
003364  000176                          .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
003365                          CR.GCR: MSG     GCR,1,CR,12.,170        ; GET CHARACTERISTICS
003365  003452                          .WORD GCR                       ;ADDRESS OF COMMAND
003366  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
003367  003503                          .WORD CR                        ;ADDRESS OF REPLY
003370  000014                          .WORD 12.                       ;SIZE OF REPLY IN WORDS
003371  000170                          .WORD 170                       ; SUCCESSFUL COMPLETION CODE
003372                          CR.CLR: MSG     DRC,2,ST,7,COMPLT       ; DRIVE CLEAR
003372  003450                          .WORD DRC                       ;ADDRESS OF COMMAND
003373  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
003374  003474                          .WORD ST                        ;ADDRESS OF REPLY
003375  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
003376  000176                          .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
        003376                  SNDONE  =       .-1             ;PREVIOUS COMMANDS CAN BE SENNT TO AN OFFLINE DRIVE
003377                          CR.DIS: MSG     DIS,2,ST,7,COMPLT       ; DISCONNECT
003377  003446                          .WORD DIS                       ;ADDRESS OF COMMAND
003400  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
003401  003474                          .WORD ST                        ;ADDRESS OF REPLY
003402  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
```

```
                003403  000176                  .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
 23             003404                  CR.SCR: MSG     SCR,2,SUB,21.,167       ; GET SUBUNIT CHARACTERISTICS
                003404  003453                  .WORD SCR                       ;ADDRESS OF COMMAND
                003405  000002                  .WORD 2                 ;SIZE OF COMMAND IN BYTES
                003406  003516                  .WORD SUB                       ;ADDRESS OF REPLY
                003407  000025                  .WORD 21.                       ;SIZE OF REPLY IN WORDS
                003410  000167                  .WORD 167                       ; SUCCESSFUL COMPLETION CODE
 24             003411                  CR.GST: MSG     GST,1,ST,7,366          ; GET STATUS
                003411  003455                  .WORD GST                       ;ADDRESS OF COMMAND
                003412  000001                  .WORD 1                 ;SIZE OF COMMAND IN BYTES
                003413  003474                  .WORD ST                        ;ADDRESS OF REPLY
                003414  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
                003415  000366                  .WORD 366                       ; SUCCESSFUL COMPLETION CODE
 25             003416                  CR.MOD: MSG     MOD,3,ST,7,COMPLT       ; MODE
                003416  003464                  .WORD MOD                       ;ADDRESS OF COMMAND
                003417  000003                  .WORD 3                 ;SIZE OF COMMAND IN BYTES
                003420  003474                  .WORD ST                        ;ADDRESS OF REPLY
                003421  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
                003422  000176                  .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
 26             003423                  CR.SEK: MSG     INS,6,ST,7,COMPLT       ; INITIATE SEEK
                003423  003460                  .WORD INS                       ;ADDRESS OF COMMAND
                003424  000006                  .WORD 6                 ;SIZE OF COMMAND IN BYTES
                003425  003474                  .WORD ST                        ;ADDRESS OF REPLY
                003426  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
                003427  000176                  .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
 27             003427                  LONG = .-1                              ; LONG TIMEOUT COMMANDS FOLLOW
 28             003430                  CR.INR: MSG     INR,1,ST,7,COMPLT       ; INITIATE RECALIBRATE
                003430  003457                  .WORD INR                       ;ADDRESS OF COMMAND
                003431  000001                  .WORD 1                 ;SIZE OF COMMAND IN BYTES
                003432  003474                  .WORD ST                        ;ADDRESS OF REPLY
                003433  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
                003434  000176                  .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
 29             003435                  CR.RUN: MSG     RUN,1,ST,7,COMPLT       ; RUN
                003435  003456                  .WORD RUN                       ;ADDRESS OF COMMAND
                003436  000001                  .WORD 1                 ;SIZE OF COMMAND IN BYTES
                003437  003474                  .WORD ST                        ;ADDRESS OF REPLY
                003440  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
                003441  000176                  .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
 30
 31                                     ;       LEVEL 2 COMMAND MESSAGE DATA STRUCTURES
 32
 33             003442     000     213  ONL:    .BYTE   0,213                   ; BRING DRIVE ONLINE
 34             003443  000012          .WORD   10.
 35             003444     000     003  DIA:    .BYTE   0,3                     ; DIAGNOSE
 36             003445  000000          .WORD   0
 37             003446     000     204  DIS:    .BYTE   0,204                   ; DISCONNECT
 38             003447  000000          .WORD   0
 39             003450     000     005  DRC:    .BYTE   0,DRVCLR                ; DRIVE CLEAR
 40             003451  000000          ERRORS: .WORD   0                       ; ERROR FLAGS
 41             003452     000     207  GCR:    .BYTE   0,GETCHR                ; GET CHARACTERISTICS
 42             003453     000     210  SCR:    .BYTE   0,GETSUB                ; GET SUBUNIT CHARACTERISTICS
 43             003454  000000          SUBUNT: .WORD   0                       ; SUBUNIT SELECTION IN LOW ORDER BYTE
 44             003455     000     011  GST:    .BYTE   0,GETSTA                ; GET STATUS
 45             003456     000     014  RUN:    .BYTE   0,14                    ; SPIN UP DRIVE
 46             003457     000     216  INR:    .BYTE   0,IRECLB                ; INITIATE RECALIBRATE
 47             003460     000     012  INS:    .BYTE   0,INSEEK                ; INITIATE SEEK
 48             003461  000000          .WORD   0                               ; INS CYLINDER/HEAD ARGUMENTS
```

```
     49 003462  000000                          .WORD   0
     50 003463  000000                          .WORD   0
     51 003464     000      201        MOD:     .BYTE   0,201                   ; CHANGE MODE
     52 003465  000000                 MODE:    .WORD   0
     53 003466     362      377        MODE1:   .BYTE   362,377
     54 003467     006      377        MODE2:   .BYTE   6,377
     55
     56 003470     000      014        RUNCMD:  .BYTE   0,14
     57          000014                RUNLBC   =       14                      ; RUN COMMAND
     58
     59 003471  000000                 COPY:    .WORD   0
     60 003472  000000                 RETRY:   .WORD   0
     61 003473  000000                 SERRTY:  .WORD   0                       ; RETRY COUNT
     62
     63                                 ;        RESPONSE  MESSAGE DATA BUFFERS
     64
     65 003474                          ST:      .BLKW   7                       ; STATUS MESSAGE BUFFER
     66 003503                          CR:      .BLKW   31.                     ; CHARACTERISTICS MESSAGE BUFF
     67                                 ;        ASSUME  SUB,CR+11.
     68
     69 003542  000000                 DMSDI:   .WORD   0                       ; DUMMY SDI CONTROL BLK
     70 003543  000000                 NSCSL:   .WORD   0                       ; # OF SECTORS IN HEADER SEARCH
     71 003544  003511                          .WORD   SUB-5                   ; POINTER TO SUBUNIT CHAR.
     72 003545                                  .BLKW   5                       ; WORD DMSDI+7 IS CLOBBERED BY UDA
     73                                                                         ; SO SET ASIDE SPACE
     74 003552                                  .BLKW   2                       ; RESERVED FOR UDA PRIMARY REVECTORING
     75 003554                                  .BLKW   8.                      ; SKIP SPACE TO POINTER
     76 003564  003536                          .WORD   DMSDI-4                 ; PRI REV INFO WRITTEN IN DMSDI+8 AND 9
     77
     78                                 ;        DISK LOCATION POINTERS
     79
     80 003565                          ROFDBN:  .BLKW   2                       ; FIRST READ ONLY DBN
     81 003567                          ROFDC:   .BLKW   3                       ; FIRST READ ONLY CYL AND GROUP
     82 003572                          LCDBN:   .BLKW   2                       ; FIRST FACTORY FORMATTED DBN
     83 003574                          LDIACYL: .BLKW   3                       ; FACTORY FORMATTED CYLINDER
     84 003577                          FDIACYL: .BLKW   2                       ; FIRST DIAGNOSTIC CYLINDER
     85 003601                          FXBNCYL: .BLKW   2                       ; FIRST XBN CYLINDER
     86 003603                          SECTRK:  .BLKW   1                       ; SECTORS PER TRACK
     87
     88 003604                          TSTBLK:  .BLKW   2                       ; TEST BLOCK NUMBER
     89 003606                          TSTCYL:  .BLKW   3                       ; TEST CYLINDER NUMBER
     90 003611                          BLOCKT:  .BLKW   1                       ; BLOCKS ON CURRENT TRACK
     91 003612                          BLOCKG:  .BLKW   1                       ; BLOCKS ON CURRENT GROUP
     92 003613                          BLOCKC:  .BLKW   1                       ; BLOCKS ON CURRENT CYL
     93 003614                          CURBLK:  .BLKW   2                       ; CURRENT BLOCK NUMBER
     94 003616                          SECGRP:  .BLKW   1                       ; NUMBER OF SECTORS PER GROUP
     95 003617                          SECCYL:  .BLKW   1                       ; NUMBER OF SECTORS PER CYL
     96 003620                          CURGRP:  .BLKW   1                       ; CURRENT GROUP NUMBER
     97 003621  000000                 CURTRK:  .WORD   0                       ; CURRENT TRACK
     98 003622  000000                 CURPAT:  .WORD   0                       ; CURRENT PATTERN
     99 003623  000000                 SECCNT:  .WORD   0                       ; SECTOR COUNT
    100
    101 003624  000000                 AREA:    .WORD   0                       ; TO STORE AREA LETTER L OR X (BN)
    102 003625  000000                 LETTER:  .WORD   0                       ; TO STORE CURRENT AREA L, D OR X (BN)
    103
    104 003626  000000                 ERRPOS:  .WORD   0
    105 003627  000000                 WFLAG:   .WORD   0                       ; FOR WRITE PROTECTION TEST
```

```
    106
    107 003630  000000                FLAG:   .WORD   0               ; TEST ALL CYLINDER FLAG
    108 003631  000000                GRPCNT: .WORD   0               ; COUNT OF CYL'S HANDLED
    109 003632  000000                OLDGRP: .WORD   0               ; HOLDS OLD GROUP VALUE FROM EVL COMPUTE
    110
    111 003633  000000  000000        VAR1:   .WORD   0,0             ; FOR CONVERT XFC
    112 003635  000000  000000        VAR2:   .WORD   0,0
    113 003637  000000                VAR3:   .WORD   0
    114 003640  000000                VAR4:   .WORD   0
    115 003641  000000  000000        CYLLO:  .WORD   0,0
    116 003643  000000                GROUP:  .WORD   0
    117 003644  000000                TRACK:  .WORD   0
    118 003645  000000                SSCTOR: .WORD   0
    119 003646  000000                INDEXS: .WORD   0
    120
    121                               ;       WRITE DATA BUFFERS
    122
    123 003647                        CHAIN:                          ; READ AND WRITE CHAIN AREA
    127 003647  000000                        .WORD   0               ; NEXT CHAIN POINTER
    131 003650  000000                        .WORD   0               ; STATUS AREA
    132 003651  000000                        .WORD   0               ; BUFFER POINTER
    133 003652  000000                        .WORD   0               ; LO ORDER HEADER
    134 003653  000000                        .WORD   0               ; HI ORDER HEADER
    135 003654  000000                        .WORD   0               ; REAL TIME COMMAND AND TRACK NUMBER
    136 003655  003542                        .WORD   DMSDI           ; POINTER TO DUMMY SDI CONTROL BLOCK
    137
```

```
      1                                          .SBTTL  DATA PATTERNS
      2 003656  000004                 PATPTR: .WORD   4                       ; FOUR PATTERNS WILL BE TESTED
      3 003657  003663                         .WORD   PAT4
      4 003660  003704                         .WORD   PAT5
      5 003661  003725                         .WORD   PAT6
      6 003662  003746                         .WORD   PAT8
      7                                  ;
      8                                  ;       THE DATA PATTERNS (LENGTH OF PATTERN IN WORDS, FOLLOWED BY PATTERN)
      9                                  ;
     10 003663  000016                 PAT4:   .WORD   16                      ; SHIFTING ONES
     11 003664  000001                         .WORD   000001
     12 003665  000003                         .WORD   000003
     13 003666  000007                         .WORD   000007
     14 003667  000017                         .WORD   000017
     15 003670  000037                         .WORD   000037
     16 003671  000077                         .WORD   000077
     17 003672  000177                         .WORD   000177
     18 003673  000377                         .WORD   000377
     19 003674  000777                         .WORD   000777
     20 003675  001777                         .WORD   001777
     21 003676  003777                         .WORD   003777
     22 003677  007777                         .WORD   007777
     23 003700  017777                         .WORD   017777
     24 003701  037777                         .WORD   037777
     25 003702  077777                         .WORD   077777
     26 003703  177777                         .WORD   177777
     27 003704  000016                 PAT5:   .WORD   16                      ; SHIFTING ZEROS
     28 003705  177776                         .WORD   177776
     29 003706  177774                         .WORD   177774
     30 003707  177770                         .WORD   177770
     31 003710  177760                         .WORD   177760
     32 003711  177740                         .WORD   177740
     33 003712  177700                         .WORD   177700
     34 003713  177600                         .WORD   177600
     35 003714  177400                         .WORD   177400
     36 003715  177000                         .WORD   177000
     37 003716  176000                         .WORD   176000
     38 003717  174000                         .WORD   174000
     39 003720  170000                         .WORD   170000
     40 003721  160000                         .WORD   160000
     41 003722  140000                         .WORD   140000
     42 003723  100000                         .WORD   100000
     43 003724  000000                         .WORD   000000
     44 003725  000016                 PAT6:   .WORD   16                      ; ALTERNATING ZERO WORD AND ONE WORD IN
     45 003726  000000                         .WORD   000000                  ; 3-2-1-1-1 SEQUENCE
     46 003727  000000                         .WORD   000000
     47 003730  000000                         .WORD   000000
     48 003731  177777                         .WORD   177777
     49 003732  177777                         .WORD   177777
     50 003733  177777                         .WORD   177777
     51 003734  000000                         .WORD   000000
     52 003735  000000                         .WORD   000000
     53 003736  177777                         .WORD   177777
     54 003737  177777                         .WORD   177777
     55 003740  000000                         .WORD   000000
     56 003741  177777                         .WORD   177777
     57 003742  000000                         .WORD   000000
```

DATA PATTERNS

```
     58 003743  177777                        .WORD   177777
     59 003744  000000                        .WORD   000000
     60 003745  177777                        .WORD   177777
     61 003746  000016              PAT8:     .WORD   16                      ; B'0101010101010101' AND
     62 003747  052525                        .WORD   052525                  ; B'1010101010101010'
     63 003750  052525                        .WORD   052525                  ; IN 3-2-1-1-1 SEQURNCE
     64 003751  052525                        .WORD   052525
     65 003752  125252                        .WORD   125252
     66 003753  125252                        .WORD   125252
     67 003754  125252                        .WORD   125252
     68 003755  052525                        .WORD   052525
     69 003756  052525                        .WORD   052525
     70 003757  125252                        .WORD   125252
     71 003760  125252                        .WORD   125252
     72 003761  052525                        .WORD   052525
     73 003762  125252                        .WORD   125252
     74 003763  052525                        .WORD   052525
     75 003764  125252                        .WORD   125252
     76 003765  052525                        .WORD   052525
     77 003766  125252                        .WORD   125252
     78
     79
     80 003767                      OBUFF:  .BLKW   257.                    ; WRITE DATA BUFFER
     81         000400              BUFFSZ = 256.
     82
     83                             ;       READ DATA BUFFERS
     84
     85 004370                      RBUF0:                                  ; FIRST BUFFER
     89 004370  100000                        .WORD   RSTOP
     93 004371  100000                        .WORD   RSTOP                   ; STATUS AND LINK POINTER
     94 004372  004377                        .WORD   RBUFD                   ; POINTER TO DATA
     95 004373                                .BLKW   2
     96 004375  013400                        .WORD   RREAL                   ; LEVEL 1 SDI COMMAND
     97 004376  003542                        .WORD   DMSDI                   ; POINTER TO DUMMY SDI CONTROL BLOCK
     98
```

```
                                        .SBTTL  START, GETU<POLL ALL PORTS>, RBUFD, & FCHAIN

                                .IF     GE,<.+<255.*3>>-HIMEM
                                        .ERROR
                                .ENDC

004377                          RBUFD:: .BLKW   274.                    ; READ DATA BUFFER

                                ;       FORMAT CHAIN

004377                          FCHAIN:
                                ;       .REPT   255.                    ; UP TO 240+ DBNS/TRK MAXIMUM
                                ;       .WORD   0                       ; BUFFER POINTER
                                ;       .WORD   0                       ; LO ORDER DBN
                                ;       .WORD   0                       ; HI ORDER DBN
                                ;       .ENDR
                                ;       .WORD   0                       ; EXTRA WORD FOR END-OF-CHAIN FLAG
004377                          START:
004377                          GETU:
                                        .SBTTL  GETU   - POLL ALL PORTS, THEN GET UNITS TO TEST
                                ;
                                ;       GET THE UNITS TO TEST
                                ;
                                ;
                                ;       POLL ALL PORTS AND FILL IN A UDA PORT INFORMATION TABLE (UNITS)
                                ;

                                ; *** NOW DO REGULAR GETU
004377  104205  000001                  MOV     #1,R5                   ; MOVE INITIAL MASK TO R5
004401  104204  003335                  MOV     #UNITS,R4               ; R4 POINTS TO UNIT TABLE
004403                          5$:     PUSH    R4                      ; SAVE R4
004403  100464                                                                  MOV R4,-(SP)
004404  104052                          MOV     R5,R2                   ; MOVE MASK TO R2
004405                                  CALL    RDSTAT                  ; GET DRIVE'S STATUS
004405  020000  001432                  ^O20000,RDSTAT
004407  115002                          TST     R2                      ; SEE IF ERROR
004410                                  BEQ     10$                     ; IF NOT, BRANCH
004410  010000  004420                  ^O10000,10$
004412  104203  003052                  MOV     #SER10,R3               ; NO DRIVE ATTACHED
004414  100643  000001                  MOV     R3,1(R4)                ; SAVE ERROR MESSAGE
004416                                  BR      85$                     ; REPORT
004416  000000  004641                  ^O0,85$
004420  114003                  10$:    CLR     R3                      ; SET UP TIMEOUT COUNT
004421  104052                  15$:    MOV     R5,R2                   ; MOVE MASK TO R2
004422                                  CALL    RDSTAT                  ; GET STATUS
004422  020000  001432                  ^O20000,RDSTAT
004424  102201  000001                  BIT     #RCVRDY,R1              ; SEE IF RECEIVER READY ASSERTED
004426                                  BNE     20$                     ; IF SO, BRANCH
004426  050000  004441                  ^O50000,20$
004430  117403                          DEC     R3                      ; DECREMENT COUNT
004431                                  BNE     15$                     ; IF INCOMPLETE, BRANCH
004431  050000  004421                  ^O50000,15$
004433  104203  003065                  MOV     #SER11,R3               ; RECEIVER READY NEVER ASSERTED
004435  100643  000001                  MOV     R3,1(R4)                ; SAVE ERROR MESSAGE
004437                                  BR      85$                     ; REPORT
004437  000000  004641                  ^O0,85$
004441  102201  000100          20$:    BIT     #AVAIL,R1               ; SEE IF DRIVE IS AVAILABLE
```

```
     76 004443                                    BNE     25$             ; IF SO, BRANCH
        004443  050000  004453                    ^050000,25$
     77 004445  104203  003360                    MOV     #SER40,R3       ; GET SECONDARY ERROR
     78 004447  100643  000001                    MOV     R3,1(R4)        ; SAVE
     79 004451                                    BR      85$             ; EXIT
        004451  000000  004641                    ^00,85$
     80 004453  104202  001750          25$:      MOV     #MAXSND,R2      ; SET UP MAXIMUM TRIES AT SENDING
     81 004455  104203  003411                    MOV     #CR.GST,R3      ; R3 POINTS TO GET STATUS COMMAND
     82 004457  104137                  30$:      MOV     (R3),R0         ; SET ADR OF SDI COMMAND BUFFER
     83 004460  104631  000001                    MOV     1(R3),R1        ; SET BUFFER LENGTH
     84 004462                                    PUSH    R2              ; SAVE R2
        004462  100462                                                                      MOV R2,-(SP)
     85 004463  104052                            MOV     R5,R2           ; SETUP FOR SEND
     86 004464  060004                            XFC     SEND            ; SEND COMMAND
     87 004465                                    POP     R2              ; RESTORE COUNT
        004465  104262                                                                      MOV (SP)+,R2
     88 004466  115001                            TST     R1              ; DID UNIT ACCEPT COMMAND
     89 004467                                    BEQ     35$             ; IF SO, BRANCH
        004467  010000  004502                    ^010000,35$
     90 004471  117402                            DEC     R2              ; DECREMENT COUNT
     91 004472                                    BNE     30$             ; IF UNEXPIRED, BRANCH
        004472  050000  004457                    ^050000,30$
     92 004474  104203  003103                    MOV     #SER12,R3       ; GET ERROR NUMBER
     93 004476  100643  000001                    MOV     R3,1(R4)        ; SAVE
     94 004500                                    BR      85$
        004500  000000  004641                    ^00,85$
     95 004502                          35$:      PUSH    R4              ; SAVE R4
        004502  100464                                                                      MOV R4,-(SP)
     96 004503  104204  000003                    MOV     #3,R4           ; SET UP SHORT TIMEOUT
     97 004505  104207  003474          40$:      MOV     #ST,R0          ; SET DATA BUFFER ADDRESS
     98 004507  104631  000003                    MOV     3(R3),R1        ; SET BUFFER LENGTH
     99 004511  104052                            MOV     R5,R2           ; SETUP FOR RECEIVE
    100 004512  060005                            XFC     RCV             ; RECEIVE SDI COMMAND
    101 004513  115001                            TST     R1              ; DID ERROR OCCUR
    102 004514                                    BEQ     70$             ; IF NOT, BRANCH
        004514  010000  004567                    ^010000,70$
    103 004516  106201  000001                    CMP     #1,R1           ; SEE IF TIMEOUT
    104 004520                                    BNE     45$             ; IF NOT, BRANCH
        004520  050000  004532                    ^050000,45$
    105 004522  117404                            DEC     R4              ; DECREMENT TIMEOUT VALUE
    106 004523                                    BNE     40$             ; IF NOT TIMEOUT, BRANCH
        004523  050000  004505                    ^050000,40$
    107 004525                                    POP     R4              ; RESTORE R4
        004525  104264                                                                      MOV (SP)+,R4
    108 004526  104203  003115                    MOV     #SER13,R3       ; GET ERROR NUMBER
    109 004530                                    BR      65$             ; BRANCH TO EXIT
        004530  000000  004563                    ^00,65$
    110 004532                          45$:      POP     R4              ; RESTORE R4
        004532  104264                                                                      MOV (SP)+,R4
    111 004533  110601                            ROR     R1              ; ROTATE INTO POSITION TO TEST
    112 004534  110601                            ROR     R1              ; SEE IF FIRST WORD NOT START FRAME
    113 004535                                    BCC     50$             ; IF NOT, BRANCH
        004535  040000  004543                    ^040000,50$
    114 004537  104203  003130                    MOV     #SER14,R3       ; GET ERROR NUMBER
    115 004541                                    BR      65$             ; BRANCH TO END OF LOOP
        004541  000000  004563                    ^00,65$
    116 004543  110601                  50$:      ROR     R1              ; SEE IF FRAMING ERROR
```

```
    117 004544                                      BCC     55$             ; IF NOT, BRANCH
        004544  040000  004552                      ^040000,55$
    118 004546  104203  003156                      MOV     #SER15,R3       ; GET ERROR NUMBER
    119 004550                                      BR      65$             ; BRANCH TO END OF LOOP
        004550  000000  004563                      ^00,65$
    120 004552  110601                  55$:        ROR     R1              ; SEE IF CHECKSUM ERROR
    121 004553                                      BCC     60$             ; IF NOT, BRANCH
        004553  040000  004561                      ^040000,60$
    122 004555  104203  003200                      MOV     #SER16,R3       ; GET ERROR NUMBER
    123 004557                                      BR      65$             ; BRANCH TO END OF LOOP
        004557  000000  004563                      ^00,65$
    124 004561  104203  003223          60$:        MOV     #SER17,R3       ; GET ERROR NUMBER
    125 004563  100643  000001          65$:        MOV     R3,1(R4)        ; SAVE
    126 004565                                      BR      85$             ; BRANCH TO END OF LOOP
        004565  000000  004641                      ^00,85$
```

```
                                        ;
                                        ;       NOW FILL IN THE TABLE WITH ALL THE SUBUNIT NUMBERS
                                        ;
004567                          70$:    POP     R4              ; RESTORE R4
004567  104264                                                                  MOV (SP)+,R4
004570  104207  177777                 MOV     #-1,R0          ; GET 'NO UNITS' FLAG
004572  100647  000001                 MOV     R0,1(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
004574  100647  000002                 MOV     R0,2(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
004576  104307  003474                 MOV     ST,R0           ; R0 HAS UNIT NUMBER
004600  104072                         MOV     R0,R2           ; COPY R0 TO R2
004601  103207  170000                 BIC     #^CHBHINB,R0    ; R0 HAS UNIT NUMBER
004603                                 PUSH    <R1,R2>         ; SAVE R1 AND R2
004603  100461                                                                  MOV R1,-(SP)
004604  100462                                                                  MOV R2,-(SP)
004605  104052                         MOV     R5,R2           ; MOVE UDA PORT MASK TO R2
004606                                 CALL    RDSTAT          ; GET STATUS
004606  020000  001432                 ^O20000,RDSTAT
004610  102201  000002                 BIT     #ATTN,R1        ; SEE IF SPINABLE
004612                                 BNE     75$             ; IF SO, BRANCH
004612  050000  004616                 ^O50000,75$
004614  101207  010000                 BIS     #10000,R0       ; SET 'NOT SPINABLE' FLAG
004616                          75$:    POP     <R2,R1>         ; RESTORE
004616  104262                                                                  MOV (SP)+,R2
004617  104261                                                                  MOV (SP)+,R1
004620  101207  040000                 BIS     #40000,R0       ; FLAG UNIT AS NOT TESTED
004622  110702                         SWAB    R2              ; SWAP R2'S BYTES
004623  110602                         ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
004624  110602                         ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
004625  110602                         ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
004626  110602                         ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
004627  103202  177760                 BIC     #LBLONB,R2      ; CLEAR ALL BUT SUBUNIT BITS
004631  100147                         MOV     R0,(R4)         ; SET UNIT NUMBER IN UNITS
004632  110602                  80$:    ROR     R2              ; MOVE SUBUNIT BIT TO CARRY
004633                                 BCC     85$             ; IF NO MORE SUBUNITS, BRANCH
004633  040000  004641                 ^O40000,85$
004635  100247                         MOV     R0,(R4)+        ; MOVE SUBUNIT NUMBER TO TABLE
004636  115407                         INC     R0              ; INCREMENT SUBUNIT NUMBER
004637                                 BR      80$             ; BRANCH
004637  000000  004632                 ^O00,80$

004641                          85$:    POP     R4              ; R4 POINTS TO START OF UNIT JUST HANDLED
004641  104264                                                                  MOV (SP)+,R4
004642  105204  000004                 ADD     #4,R4           ; R4 WILL POINT TO NEXT UNIT (CLEAR CARRY FOR ROL)
004644  110205                         ROL     R5              ; R5 HAS NEXT UNIT PORT MASK
004645  106205  000020                 CMP     #20,R5          ; SEE IF ALL PORTS TESTED
004647                                 BNE     5$              ; IF NOT, BRANCH
004647  050000  004403                 ^O50000,5$
```

```
                                        ;
                                        ;       NOW GET THE PLUG NUMBERS TO TEST AND FIND THEM IN THE TABLE
                                        ;
004651  104207  060012                  MOV     #UTOTST,R0      ; GET WHAT SUBUNIT NUMBERS TO TEST REQUEST
004653                                  CALL    HOSTRQ          ; GET THE PLUG NUMBERS
004653  020000  001463                  ^O20000,HOSTRQ
004655  104207  001521                  MOV     #OUT.01,R0      ; R0 POINTS TO UNIT NUMBERS TO TEST
004657  104201  003335          90$:    MOV     #UNITS,R1       ; R1 POINTS TO UDA PORT INFORMATION
004661  104112                  95$:    MOV     (R1),R2         ; R2 HAS UNIT
004662                                  BMI     115$            ; IF NONE, BRANCH
004662  070000  004712                  ^O70000,115$
004664                                  PUSH    R1              ; SAVE R1
004664  100461                                                                  MOV R1,-(SP)
004665  103202  160000          100$:   BIC     #160000,R2      ; CLEAR 'NOT TESTED', LEAVE 'UNSPINABLE' SET
004667  106172                          CMP     (R0),R2         ; SEE IF IT IS A UNIT TO TEST
004670                                  BNE     105$            ; NO MATCH
004670  050000  004676                  ^O50000,105$
004672  100112                          MOV     R2,(R1)         ; SAVE UNIT AS ONE TO TEST
004673                                  POP     R1              ; RESTORE STACK
004673  104261                                                                  MOV (SP)+,R1
004674                                  BR      160$            ; LOOK FOR THE NEXT ONE
004674  000000  005032                  ^O0,160$
004676  115401          105$:   INC     R1              ; POINT TO NEXT SUBUNIT
004677  104012                  MOV     R1,R2           ; COPY TO R2
004700  107202  003335          SUB     #UNITS,R2       ; SUBTRACT STARTING ADDRESS
004702  102202  000003          BIT     #3,R2           ; SEE IF STILL ON SAME UNIT
004704                          BEQ     110$            ; IF NOT, BRANCH
004704  010000  004711          ^O10000,110$
004706  104112                  MOV     (R1),R2         ; SEE IF ANY MORE SUBUNITS
004707                          BPL     100$            ; IF SO, BRANCH
004707  030000  004665          ^O30000,100$
004711          110$:   POP     R1              ; RESTORE R1
004711  104261                                                                  MOV (SP)+,R1
004712  105201  000004          115$:   ADD     #4,R1           ; LOOK AT NEXT UNIT
004714  106201  003355          CMP     #UNITS+16.,R1   ; SEE IF ENTIRE TABLE SEARCHED
004716                          BNE     95$             ; IF NOT, BRANCH
004716  050000  004661          ^O50000,95$
```

```
                                        ;
                                        ;       DIDN'T FIND THE REQUESTED UNITS -- DUMP ALL KNOWLEDGE OF THE
                                        ;       UDA PORTS AND DIE
                                        ;
004720  104170  001523                  MOV     (R0),OUT.03     ; SAVE UNIT NUMBER IN REQUEST BUFFER
004722  104204  001525                  MOV     #OUT.05,R4      ; R4 POINTS TO OUTPUT BUFFER
004724  104205  003335                  MOV     #UNITS,R5       ; R5 POINTS TO UNIT TABLE
004726  104157                  120$:   MOV     (R5),R0         ; GET FIRST WORD OF UNIT
004727                                  BPL     125$            ; IF VALID UNIT WAS FOUND, BRANCH
004727  030000  004736                  ^O030000,125$
004731  104657  000001                  MOV     1(R5),R0        ; GET POINTER TO ERROR MESSAGE
004733  100247                          MOV     R0,(R4)+        ; SAVE IN OUTPUT BUFFER
004734                                  BR      155$            ; EXIT
004734  000000  004776                  ^O00,155$
004736                          125$:   PUSH    R5              ; SAVE POINTER TO UNIT TABLE
004736  100465                                                                  MOV R5,-(SP)
004737  102207  010000                  BIT     #10000,R0       ; SEE IF DRIVE UNSPINABLE
004741                                  BEQ     130$            ; IF SPINABLE, BRANCH
004741  010000  004747                  ^O010000,130$
004743  104207  003402                  MOV     #SER41,R0       ; REPORT DRIVE(S) UNSPINABLE
004745                                  BR      135$            ; BRANCH
004745  000000  004751                  ^O00,135$
004747  104207  003255          130$:   MOV     #SER18,R0       ; GET POINTER TO ERROR MESSAGE
004751  100247                  135$:   MOV     R0,(R4)+        ; SAVE
004752  114007                          CLR     R0              ; CLEAR COUNT
004753  104251                  140$:   MOV     (R5)+,R1        ; GET UNIT NUMBER
004754                                  BMI     145$            ; IF INVALID, BRANCH
004754  070000  004763                  ^O070000,145$
004756  115407                          INC     R0              ; INCREMENT COUNT
004757  106207  000004                  CMP     #4,R0           ; SEE IF MAX
004761                                  BNE     140$            ; IF NOT, LOOP
004761  050000  004753                  ^O050000,140$
004763  104671  002260          145$:   MOV     SER18E-1(R0),R1 ; GET POINTER TO CORRECT ERROR MESSAGE
004765  100241                          MOV     R1,(R4)+        ; MOVE INTO OUTPUT BUFFER
004766                                  POP     R5              ; RESTORE R5
004766  10426[                                                                  MOV (SP)+,R5
004767                                  PUSH    R5              ; SAVE R5
004767  100465                                                                  MOV R5,-(SP)
004770  104251                  150$:   MOV     (R5)+,R1        ; GET SUBUNIT NUMBER
004771  100241                          MOV     R1,(R4)+        ; SAVE
004772  117407                          DEC     R0              ; DECREMENT COUNT
004773                                  BNE     150$            ; IF COUNT INCOMPLETE, BRANCH
004773  050000  004770                  ^O050000,150$
004775                                  POP     R5              ; RESTORE R5
004775  104265                                                                  MOV (SP)+,R5
004776  105205  000004          155$:   ADD     #4,R5           ; POINT TO NEXT UNIT TABLE
005000  106205  003355                  CMP     #UNITS+16.,R5   ; SEE IF ENTIRE TABLE SEARCHED
005002                                  BNE     120$            ; IF NOT, BRANCH
005002  050000  004726                  ^O050000,120$
005004                                  DEVFTL  2000            ; REPORT FATAL ERROR (WILL SHOW UP AS A 5000)
005004  104200  002734  001524                                                  MOV     #MS2000,OUT.04
005007  104300  002212  001523                                                  MOV     LUNIT,OUT.03
005012  104202  051610                                                          MOV     #2000!FTLDEV+3000.,R2
005014  104020  001522                                                          MOV     R2,OUT.02
005016  104200  005016  001521                                                  MOV     #.,OUT.01
005021  104200  060013  001520                                                  MOV     #ERRMES,OUT.RQ
005024  104307  001520                  MOV     OUT.RQ,R0       ; SET UP FOR REPORT
```

```
     40 005026                                  CALL    HOSTRQ          ; REPORT ERROR
        005026  020000  001463                  ^O020000,HOSTRQ
     41 005030                                  BR      DONECD          ; END TEST
        005030  000000  002347                  ^O0,DONECD
     42
     43 005032  115407                  160$:   INC     R0              ; POINT TO NEXT UNIT TO TEST
     44 005033  104172                          MOV     (R0),R2         ; CHECK NEXT UNIT
     45 005034                                  BPL     90$             ; FIND IN UDA PORT INFORMATION
        005034  030000  004657                  ^O030000,90$
     46 005036                                  BR      T3STRT          ; START TEST
        005036  000000  002265                  ^O0,T3STRT
     47 005040                          ENDGTU:
     48                                         .SBTTL  REST OF FORMAT CHAIN
     49
     50 005040                                  .BLKW   <<255.*3>-<ENDGTU-GETU>>
     51
     52          005774                         ENDPNT = .
     53
     54                                 ;       OVERLAYS START HERE!!!
     55
     56          005774                 OVRLPT  =       .
     57
```

```
                                        .SBTTL  INIT DRIVE AND LOOK AT DRIVE SIGNALS

                                ;START OF TEST CODE
                                ;
                                ;INPUTS:
                                ;       LUNIT - LOGICAL UNIT NUMBER OF DRIVE UNDER TEST
                                ;       SDI - SDI INTERCONNECT CODE FOR DRIVE

                                ;INITIALIZE THE DRIVE

005774  104302  003356      STRTST: MOV SDI,R2                      ;GET SDI SELECT CODE
005776  060011                      XFC DINIT                       ;INITIALIZE THE DRIVE

                                ;WAIT FOR DRIVE TO ASSERT RECEIVER READY
                                ;TIME OUT AFTER ...?

005777  104201  001400              MOV     #1400,R1                ; SETUP TIMEOUT
006001  117401              1$:     DEC     R1                      ; DECREMENT DELAY
006002                              BNE     1$                      ; LOOP UNTIL DONE
006002  050000  006001              ^O50000,1$
006004  114005                      CLR     R5                      ; GET TIMEOUT COUNTER
006005  104201  000310      2$:     MOV     #200.,R1                ; INNER LOOP TIMEOUT
006007  117401              3$:     DEC     R1                      ; DELAY
006010                              BNE     3$                      ;  UNTIL DONE
006010  050000  006007              ^O50000,3$
006012                              CALL    RTDS                    ; GET REAL TIME DRIVE STATE
006012  020000  001371              ^O20000,RTDS
006014  115002                      TST     R2                      ; SEE IF ERROR OCCURRED
006015                              BEQ     4$                      ; IF OK, CONTINUE
006015  010000  006023              ^O10000,4$
006017                              PUSH    R1
006017  100461                                                      MOV R1,-(SP)
006020                              CALL    TESTEW                  ; IF SO, BRANCH
006020  020000  002355              ^O20000,TESTEW
006022                              POP     R1
006022  104261                                                      MOV (SP)+,R1
006023  102201  000001      4$:     BIT     #RCVRDY,R1              ; CHECK RECEIVER READY LINES
006025                              BNE     T                       ; ADVANCE IF ASSERTED
006025  050000  006054              ^O50000,T
006027  105305                      DEC     R5                      ; DECREMENT TIME OUT COUNTER
006030                              BNE     2$                      ; IF UNEXPIRED, BRANCH
006030  050000  006005              ^O50000,2$
006032                              HARDER  9                       ;REPORT ERROR
006032  104200  000306  001524                                              MOV     #MS9,OUT.04
006035  104300  002212  001523                                              MOV     LUNIT,OUT.03
006040  104202  105701                                                      MOV     #9!ERHARD+3000.,R2
006042  104020  001522                                                      MOV     R2,OUT.02
006044  104200  006044  001521                                              MOV     #.,OUT.01
006047  104200  060013  001520                                              MOV     #ERRMES,OUT.RQ
006052                              CALL    TESTEV                  ;EXIT TESTING THIS DRIVE
006052  020000  002364              ^O20000,TESTEV
```

```
GET DRIVE CHARACTERISTICS

                                        .SBTTL  GET DRIVE CHARACTERISTICS
                                ;GET DRIVE CHARACTERISTICS

006054  104200  000003  002173  T:    MOV     #3,SDISTO               ; SET UP TEMPORARY SHORT TIMEOUT VALUE
006057  104203  003365                MOV     #CR.GCR,R3              ; POINT TO GET CHARS COMMAND
006061  104200  002630  003357        MOV     #MS.GCR,COMND
006064                                CALL    TALK                    ; SDI INTERCHANGE
006064  020000  001563                ^O020000,TALK
006066  115002                        TST     R2                      ; SEE IF ERROR OCCURRED
006067                                BEQ     1$                      ; IF NOT, BRANCH
006067  010000  006073                ^O10000,1$
006071                                CALL    TESTEX                  ; IF SO, REPORT
006071  020000  002372                ^O020000,TESTEX
006073  104307  003503        1$:     MOV     CR+SHRTTO,R0            ; GET SHORT TIMEOUT
006075  103207  177760                BIC     #LBLONB,R0              ; CLEAR UNUSED BITS
006077                                CALL    TO                      ; SET UP TIMEOUT
006077  020000  002175                ^O020000,TO
006101  104070  002173                MOV     R0,SDISTO               ; SAVE IN SHORT TIMEOUT
006103  104307  003504                MOV     CR+LONGTO,R0            ; GET LONG TIMEOUT
006105  103207  177760                BIC     #LBLONB,R0              ; CLEAR UNUSED BITS
006107                                CALL    TO                      ; SET UP TIMEOUT
006107  020000  002175                ^O020000,TO
006111  104070  002174                MOV     R0,SDILTO               ; SAVE IN LONG TIMEOUT
006113  104307  003504                MOV     CR+RCTCPS,R0
006115  110707                        SWAB    R0
006116  103207  177760                BIC     #177760,R0
006120  104070  003471                MOV     R0,COPY                 ; GET AND SAVE NUMBER OF RCT COPIES


                                        .SBTTL  ISSUE ONLINE COMMAND
                                ;ISSUE ONLINE COMMAND AND CHECK RESPONSE

006122  104203  003360                MOV     #CR.ONL,R3              ; POINT TO ONLINE COMMAND
006124  104200  002604  003357        MOV     #MS.ONL,COMND           ; SET UP FOR ERROR
006127                                CALL    TALK                    ; SDI INTERCHANGE
006127  020000  001563                ^O020000,TALK
006131  115002                        TST     R2                      ; SEE IF ERROR OCCURRED
006132                                BEQ     2$                      ; IF NOT, BRANCH
006132  010000  006136                ^O10000,2$
006134                                CALL    TESTEX                  ; IF SO, REPORT
006134  020000  002372                ^O020000,TESTEX
006136                          2$:
```

```
ISSUE ONLINE COMMAND

                                        .SBTTL  ISSUE ONLINE COMMAND
                                ;ISSUE GET STATUS COMMAND AND CHECK THAT ATTENTION DROPS
                                
006136  104203  003411                  MOV     #CR.GST,R3              ; POINT TO GET STATUS COMMAND
006140  104200  002667  003357          MOV     #MS.GST,COMND           ; SET UP FOR ERROR
006143                                  CALL    TALK                    ; SDI INTERCHANGE
006143  020000  001563                  ^O20000,TALK
006145  115002                          TST     R2                      ; SEE IF ERROR OCCURRED
006146                                  BEQ     TOA                     ; IF NOT, BRANCH
006146  010000  006152                  ^O10000,TOA
006150                                  CALL    TESTEX                  ; IF SO, REPORT
006150  020000  002372                  ^O20000,TESTEX
006152                          TOA:    CALL    RTDS                    ; GET REAL TIME DRIVE STATE
006152  020000  001371                  ^O20000,RTDS
006154  102201  000002                  BIT     #ATTN,R1                ; IS ATTENTION ASSERTED?
006156                                  BEQ     TOB                     ; COMMAND SHOULD HAVE CLEARED IT
006156  010000  006162                  ^O10000,TOB
                                ;       ERROR - ATTEN NOT DEASSERTED
006160                                  CALL    TESTEW
006160  020000  002355                  ^O20000,TESTEW
006162                          TOB:
                                
                                
                                
                                
                                
                                        .SBTTL  ISSUE DRIVE CLEAR COMMAND
                                ;ISSUE DRIVE CLEAR COMMAND
                                
006162  104200  000377  003451          MOV     #LOBYTE,ERRORS          ; MOVE TO DRIVE CLEAR SDI AREA
006165  104203  003372                  MOV     #CR.CLR,R3              ; POINT TO DRIVE CLEAR
006167  104200  002611  003357          MOV     #MS.CLR,COMND           ; SET UP FOR ERROR
006172                                  CALL    TALK                    ; SDI INTERCHANGE
006172  020000  001563                  ^O20000,TALK
006174  115002                          TST     R2                      ; SEE IF ERROR OCCURRED
006175                                  BEQ     1$                      ; IF NOT, BRANCH
006175  010000  006201                  ^O10000,1$
006177                                  CALL    TESTEX                  ; IF SO, REPORT
006177  020000  002372                  ^O20000,TESTEX
```

ISSUE CHANGE MODE COMMAND

```
                                                .SBTTL  ISSUE CHANGE MODE COMMAND
                                        ;       ISSUE CHANGE MODE COMMAND TO ENABLE DIAG CYL ACCESS
006201                                  1$:
006201  104203  003416                        MOV     #CR.MOD,R3              ; POINT TO CHANGE MODE COMMAND
006203  104300  003466  003465                MOV     MODE1,MODE              ; WRITE PROTECT, DIAG CYL ACCESS, NO FORMATTING
006206  104200  002676  003357                MOV     #MS.MOD,COMND           ; SET UP FOR ERROR
006211                                        CALL    TALK                    ; SDI INTERCHANGE
006211  020000  001563                        ^O20000,TALK
006213  115002                                TST     R2                      ; SEE IF ERROR OCCURRED
006214                                        BEQ     2$                      ; IF NOT, BRANCH
006214  010000  006220                        ^O10000,2$
006216                                        CALL    TESTEX                  ; IF SO, REPOR
006216  020000  002372                        ^O20000,TESTEX


                                                .SBTTL  SPIN UP DRIVE
                                        ;SPIN UP DRIVE
006220  104203  003435                  2$:   MOV     #CR.RUN,R3              ; POINT TO RUN COMMAND
006222  104200  002726  003357                MOV     #MS.RUN,COMND           ; SET UP FOR ERROR
006225                                        CALL    TALK                    ; SDI INTERCHANGE
006225  020000  001563                        ^O20000,TALK
006227  115002                                TST     R2                      ; SEE IF ERROR OCCURRED
006230                                        BEQ     3$                      ; IF NOT, BRANCH
006230  010000  006234                        ^O10000,3$
006232                                        CALL    TESTEX                  ; IF SO, REPORT
006232  020000  002372                        ^O20000,TESTEX
006234                                  3$:
```

```
ISSUE INITIATE RECALIBRATE COMMAND

                                        .SBTTL  ISSUE INITIATE RECALIBRATE COMMAND
                                ; ISSUE INITIATE RECALIBRATE COMMAND
                                
 006234  104203  003430                 MOV     #CR.INR,R3              ; POINT TO RECALIBRATE COMMAND
 006236  104200  002712  003357         MOV     #MS.INR,COMND           ; SET UP FOR ERROR
 006241                                 CALL    TALK                    ; SDI INTERCHANGE
 006241  020000  001563                 ^O20000,TALK
 006243  115002                         TST     R2                      ; SEE IF ERROR OCCURRED
 006244                                 BEQ     4$                      ; IF NOT, BRANCH
 006244  010000  006252                 ^O10000,4$
 006246                                 PUSH    R1
 006246  100461                                                         MOV R1,-(SP)
 006247                                 CALL    TESTEX
 006247  020000  002372                 ^O20000,TESTEX
 006251                                 POP     R1
 006251  104261                                                         MOV (SP)+,R1
 006252  104204  000004         4$:     MOV     #4,R4                   ; R4 IS DECREMENT COUNTER
 006254                         T3C:    CALL    RTDS                    ; GET DRIVE SIGNALS
 006254  020000  001371                 ^O20000,RTDS
 006256  115002                         TST     R2                      ; SEE IF ERROR
 006257                                 BNE     TESTEW                  ; IF SO, BRANCH
 006257  050000  002355                 ^O50000,TESTEW
 006261  102201  000002                 BIT     #ATTN,R1                ; DID ATTENTION SET?
 006263                                 BEQ T3D                         ;  NO
 006263  010000  006373                 ^O10000,T3D
                                ; CHECK ERROR BITS - IF SET, REPORT "AFTER RECAL ERROR BITS ARE SET" OR SOMETHING THEN EXIT
                                ; IF LOGGABLE INFO BIT SET (NO ERROR BITS) MESSAGE- LOGGABLE INFO AFTER RECAL, CONTINUE TEST
 006265                                 PUSH    R1                      ; SAVE R1
 006265  100461                                                         MOV R1,-(SP)
 006266  104203  003411                 MOV     #CR.GST,R3              ; POINT TO GET STATUS COMMAND
 006270  104200  002667  003357         MOV     #MS.GST,COMND           ; SET UP FOR ERROR
 006273                                 CALL    TALK                    ; SEND AND RECEIVE COMMAND
 006273  020000  001563                 ^O20000,TALK
 006275  115002                         TST     R2                      ; ANY ERRORS?
 006276                                 BNE     5$                      ; IF SO, EXIT
 006276  050000  006331                 ^O50000,5$
 006300  104303  003476                 MOV     ST+ST.ERR,R3            ; GET ERROR INFORMATION
 006302  102203  000350                 BIT     #<ST.FE+ST.RE+ST.PE+ST.WE>,R3 ; ANY ERRORS?
 006304                                 BEQ     T3A                     ; IF OK CONTINUE
 006304  010000  006340                 ^O10000,T3A
 006306                                 HARDER  20,#SER22
 006306  104200  001050  001524                                         MOV     #MS20,OUT.04
 006311  104200  003276  001525                                         MOV     #SER22,OUT.05
 006314  104300  002212  001523                                         MOV     LUNIT,OUT.03
 006317  104202  105714                                                 MOV     #20!ERHARD+3000.,R2
 006321  104020  001522                                                 MOV     R2,OUT.02
 006323  104200  006323  001521                                         MOV     #.,OUT.01
 006326  104200  060013  001520                                         MOV     #ERRMES,OUT.RQ
 006331                         5$:     POP     R1
 006331  104261                                                         MOV (SP)+,R1
 006332  104203  001526                 MOV     #OUT.06,R3              ; ELSE SET UP ERROR INFO
 006334                                 CALL    TESTEY
 006334  020000  002376                 ^O20000,TESTEY
 006336                                 BR      T4A
 006336  000000  006424                 ^O0,T4A
 006340  102200  000010  003475 T3A:    BIT     #ST.EL,ST+1             ; ANY LOGGABLE INFO?
 006343                                 BEQ     T3B                     ; IF NOT CONTINUE
```

ISSUE INITIATE RECALIBRATE COMMAND

```
                006343  010000  006372                  ^010000,T3B
             36 006345  104203  001524                  MOV     #OUT.04,R3          ; IF SO, SET UP POINTER
             37 006347                                  CALL    STRST               ; AND SET UP INFO
                006347  020000  002572                  ^020000,STRST
             38 006351                                  MSSG    21,#SER22
                006351  104200  001073  001522                      MOV     #MS21,OUT.02
                006354  104200  003276  001523                      MOV     #SER22,OUT.03
                006357  100467                                              MOV R0,-(SP)
                006360  100461                                              MOV R1,-(SP)
                006361  104300  002212  001521                      MOV     LUNIT,OUT.01
                006364  104207  060015                              MOV     #MESSAG,R0
                006366  020000  001463                  ^020000,HOSTRQ
                006370  104261                                              MOV (SP)+,R1
                006371  104267                                              MOV (SP)+,R0
             39 006372                          T3B:    POP     R1                  ; RESTORE R1
                006372  104261                                              MOV (SP)+,R1
             40 006373  102201  100000          T3D:    BIT     #RWRDY,R1           ; DID R/W READY SET?
             41 006375                                  BNE T4A                     ; IF SO, BRANCH
                006375  050000  006424                  ^050000,T4A
             42 006377  117404                          DEC     R4                  ; TRY AGAIN?
             43 006400                                  BNE     T3C
                006400  050000  006254                  ^050000,T3C
             44 006402                                  HARDER  16                  ;R/W READY DID NOT SET AFTER RFCALIBRATE COMMAND
                006402  104200  000635  001524                      MOV     #MS16,OUT.04
                006405  104300  002212  001523                      MOV     LUNIT,OUT.03
                006410  104202  105710                              MOV     #16!ERHARD+3000.,R2
                006412  104020  001522                              MOV     R2,OUT.02
                006414  104200  006414  001521                      MOV     #.,OUT.01
                006417  104200  060013  001520                      MOV     #ERRMES,OUT.RQ
             45 006422                                  CALL    TESTEW
                006422  020000  002355                  ^020000,TESTEW
```

```
                                                .SBTTL  GET SUBUNIT CHARACTERISTICS
                                        ;       GET SUBUNIT CHARACTERISTICS

006424  104301  003355        T4A:    MOV     UNITNB,R1               ; FIND SUBUNIT MASK FOR COMMAND
006426  103201  177774                BIC     #^C3,R1                 ; CLEAR UNUSED BITS OF INDEX
006430  104207  000010                MOV     #10,R0
006432  110207                1$:     ROL     R0                      ; ROTATE
006433  117401                        DEC     R1                      ;  UNTIL DONE
006434                                BPL     1$
006434  030000  006432                ^O30000,1$
006436  103207  177417                BIC     #LBHINB,R0              ; CLEAR UNUSABLE BITS
006440  104070  003454                MOV     R0,SUBUNT
006442  104203  003404                MOV     #CR.SCR,R3              ; POINT TO GET SUBUNIT CHARACTERISTICS
006444  104200  002647  003357        MOV     #MS.SCR,COMND           ; SET UP FOR ERROR
006447                                CALL    TALK                    ; SDI INTERCHANGE
006447  020000  001563                ^O20000,TALK
006451  115002                        TST     R2                      ; SEE IF ERROR OCCURRED
006452                                BEQ     2$
006452  010000  006456                ^O10000,2$
006454                                CALL    TESTEX
006454  020000  002372                ^O20000,TESTFX


                                                .SBTTL  SEEK TO CYLINDER 0, GROUP 0
                                        ;SEEK TO CYLINDER 0, GROUP 0

006456  104201  003[illegible]  2$:     MOV     #TSTCYL,R1
006460  114000  00[illegible]           CLR     TSTCYL                  ; LO ORDER CYL 0
006462  104300  003[illegible]  003607  MOV     SUB+LBNCYL+1,TSTCYL+1   ; HI ORDER CYL 0
006465  103200  007777  003607        BIC     #HBHINB,TSTCYL+1
006470  114000  003610                CLR     TSTCYL+2                ; GROUP 0
006472                                CALL SEEK
006472  020000  002621                ^O20000,SEEK
```

```
                                        .SBTTL  CALCULATE NUMBERS
                                ;       CALCULATE NUMBER OF SECTORS PER TRACK
                                ;       NO. OF SECTORS/TRACK =(NO. OF LBN'S/TRACK) + (NO. OF RBN'S/TRACK)

006474  104301  003522        T4A1:   MOV SUB+RBNTRK,R1               ; GET RBN'S/TRACK
006476  103201  177600                BIC #177600,R1
006500  104307  003527                MOV SUB+LBNTRK,R0               ; GET LBN'S/TRACK
006502  103207  177400                BIC #HIBYTE,R0                  ; ZERO UPPER BITS
006504  105071                        ADD R0,R1                       ; ADD NUMBER OF LBN'S TO RBN'S
006505  104010  003603                MOV R1,SECTRK                   ; SAVE
006507  105011                        ADD R1,R1                       ; COMPUTE SECTORS TIMES TWO
006510  104010  003543                MOV R1,NSCSL                    ; AS SECTORS READ BEFORE DECLARING
                                                                        ; NO HEADER FOUND

                                ;COMPUTE FIRST CYLINDER IN XBN AREA

006512  104301  003516                MOV     SUB+LBNCYL,R1
006514  104010  003601                MOV     R1,FXBNCYL              ; FIRST XBN CYLINDER (LO)
006516  104307  003517                MOV     SUB+LBNCYL+1,R0
006520  103207  170000                BIC     #^CHBHINB,R0            ;CLEAR SUBUNIT CYL BITS
006522  104070  003602                MOV     R0,FXBNCYL+1            ; FIRST XBN CYLINDER (HI)

                                ;COMPUTE FIRST CYLINDER IN DBN AREA

006524  105301  003537                ADD SUB+XBNCYL,R1               ; ADD NUMBER OF XBN CYLINDERS
006526                                BCC T4B                         ; BRANCH IF NO CARRY
006526  040000  006531                ^040000,T4B
006530  115407                        INC R0                          ; INCREMENT HI ORDER CYL
006531  104010  003577        T4B:    MOV R1,FDIACYL                  ; STORE STARTING DBN CYLINDER
006533  104070  003600                MOV R0,FDIACYL+1

                                ;COMPUTE LAST CYLINDER IN DBN AREA

006535  104303  003540                MOV     SUB+DBNCYL,R3           ; GET NUMBER OF DBN CYLINDERS
006537  110703                        SWAB    R3
006540  103203  177400                BIC     #HIBYTE,R3
006542                                BNE     T4C
006542  050000  006566                ^050000,T4C
006544                                HARDER  17                      ;CHARACTERISTICS SAY LESS THAN 1 DIAGNOSTIC CYLINDER
006544  104200  000677  001524                                                MOV     #MS17,OUT.04
006547  104300  002212  001523                                                MOV     LUNIT,OUT.03
006552  104202  105711                                                        MOV     #17!ERHARD+3000.,R2
006554  104020  001522                                                        MOV     R2,OUT.02
006556  104200  006556  001521                                                MOV     #.,OUT.01
006561  104200  060013  001520                                                MOV     #ERRMES,OUT.RQ
006564                                CALL    TESTEV
006564  020000  002364                ^020000,TESTEV
006566                          T4C:
                                ; NO LONGER NEED LAST DIAG CYL - JUST SAVE STARTING DBN, LAST TRACK, LAST GROUP, LAST CYL
                                ;               AND USE COMPUT XFC HEAVILY
006566  117403                        DEC     R3                      ; REDUCE BY ONE AND ADD TO
006567  105031                        ADD     R3,R1                   ; FIRST DBN CYLINDER TO
006570                                BCC     T4D                     ; COMPUTE LAST CYLINDER
006570  040000  006573                ^040000,T4D
006572  115407                        INC     R0
006573  104010  003574        T4D.    MOV     R1,LDIACYL              ; STORE LAST DBN CYLINDER
006575  104070  003575                MOV     R0,LDIACYL+1
```

CALCULATE NUMBERS

```
     48 006577  104303  003520                MOV     SUB+GRPCYL,R3           ; GET GROUPS/CYLINDER
     49 006601  103203  177400                BIC     #HIBYTE,R3              ; CLEAR UNUSED BITS
     50 006603  117403                        DEC     R3                      ; LAST GROUP
     51 006604  104030  003576                MOV     R3,LDIACYL+2
```

READ ALL FACTORY FORMATTED TRACKS

```
                                        .SBTTL  READ ALL FACTORY FORMATTED TRACKS
                                ;READ ALL FACTORY FORMATTED TRACKS.
                                ;REPORT ERROR IF ALL SECTORS ON A TRACK READ WITH AN ERROR.
                                
                                ;COMPUTE DBN OF FIRST BLOCK ON FACTORY FORMATTED CYLINDER
                                
006606  114005                  T6:     CLR     R5
006607  104303  003521                  MOV     SUB+TRKGRP,R3           ; TRACKS/GROUP
006611  103203  177400                  BIC     #HIBYTE,R3
006613  104304  003603                  MOV     SECTRK,R4               ; X SECTORS/TRACK
006615  105035                  T6A:    ADD     R3,R5                   ; TO COMPUTE NUMBER OF SECTORS PER GROUP
006616  117404                          DEC     R4
006617                                  BNE     T6A
006617  050000  006615                  ^O50000,T6A
006621  104050  003616                  MOV     R5,SECGRP               ; SAVE SECTORS/GROUP
006623                                  PUSH    R2                      ; SAVE R2
006623  100462                                                                  MOV R2,-(SP)
006624  104302  003520                  MOV     SUB+GRPCYL,R2           ; GET GROUPS/CYL
006626  103202  177400                  BIC     #HIBYTE,R2              ; CLEAR UNUSED BITS
006630                                  PUSH    R2                      ; SAVE R2
006630  100462                                                                  MOV R2,-(SP)
006631  114000  003617                  CLR     SECCYL
006633  105300  003616  003617  2$:     ADD     SECGRP,SECCYL           ; GET SECTORS/CYL
006636  117402                          DEC     R2
006637                                  BNE     2$
006637  050000  006633                  ^O50000,2$
006641                                  POP     R2                      ; RESTORE R2
006641  104262                                                                  MOV (SP)+,R2
006642  104301  003540                  MOV     SUB+DBNCYL,R1           ; GET CYLINDERS IN DBN AREA
006644  110701                          SWAB    R1
006645  103201  177400                  BIC     #HIBYTE,R1              ; TO COMPUTE BLOCK NUMBER OF FIRST BLOCK
006647  114005                          CLR     R5                      ; CLEAR PRODUCT
006650  105025                  1$:     ADD     R2,R5
006651  117401                          DEC     R1
006652                                  BNE     1$
006652  050000  006650                  ^O50000,1$
006654                                  POP     R2                      ; RESTORE R2
006654  104262                                                                  MOV (SP)+,R2
006655  114007                          CLR     R0
006656                                  PUSH    R5                      ; SAVE NUMBER OF GROUPS IN DIAGNOSTIC AREA
006656  100465                                                                  MOV R5,-(SP)
006657  117405                          DEC     R5                      ; SO DBN COMPUTED IS FIRST ON LAST GROUP
006660  105301  003616          T6C:    ADD     SECGRP,R1
006662                                  BCC     T6D
006662  040000  006665                  ^O40000,T6D
006664  115407                          INC     R0
006665  117405                  T6D:    DEC     R5
006666                                  BNE     T6C
006666  050000  006660                  ^O50000,T6C
006670  104010  003572                  MOV     R1,LCDBN
006672  104070  003573                  MOV     R0,LCDBN+1
006674                                  POP     R5                      ; R5 IS NUMBER OF GROUPS IN DIAG AREA
006674  104265                                                                  MOV (SP)+,R5
006675  104307  003540                  MOV     SUB+DBNCYL,R0           ; GET NUMBER OF READ ONLY GROUPS
006677  103207  177400                  BIC     #HIBYTE,R0              ; CLEAR UNUSED BITS
006701                                  BNE     1$                      ; IF READ ONLY GROUPS > 0 THEN BRANCH
006701  050000  006725                  ^O50000,1$
```

```
    46 006703                                          HARDER  14                      ; ZERO READ ONLY GROUPS
       006703  104200  000501  001524                                                          MOV     #MS14,OUT.04
       006706  104300  002212  001523                                                          MOV     LUNIT,OUT.03
       006711  104202  105706                                                                  MOV     #14!ERHARD+3000.,R2
       006713  104020  001522                                                                  MOV     R2,OUT.02
       006715  104200  006715  001521                                                          MOV     #.,OUT.01
       006720  104200  060013  001520                                                          MOV     #ERRMES,OUT.RQ
    47 006723                                          BR      TESTEV
       006723  000000  002364                          ^O0,TESTEV
    48 006725  104300  003574  003567          1$:     MOV     LDIACYL,ROFDC
    49 006730  104300  003575  003570                  MOV     LDIACYL+1,ROFDC+1
    50 006733  107075                                  SUB     R0,R5                   ; R5 HAS NUMBER OF READ/WRITE GROUPS
    51 006734                                          BEQ     2$                      ; IF R/W GROUPS ZERO, ERROR
       006734  010000  006740                          ^O10000,2$
    52 006736                                          BPL     3$                      ; IF GREATER THAN ZERO, BRANCH
       006736  030000  006762                          ^O30000,3$
    53 006740                                  2$:     HARDER  15                      ; NO READ/WRITE GROUPS
       006740  104200  000555  001524                                                          MOV     #MS15,OUT.04
       006743  104300  002212  001523                                                          MOV     LUNIT,OUT.03
       006746  104202  105707                                                                  MOV     #15!ERHARD+3000.,R2
       006750  104020  001522                                                                  MOV     R2,OUT.02
       006752  104200  006752  001521                                                          MOV     #.,OUT.01
       006755  104200  060013  0015^0                                                          MOV     #ERRMES,OUT.RQ
    54 006760                                          BR      TESTEV
       006760  000000  002364                          ^O0,TESTEV
    55 006762  104304  003520                  3$:     MOV     SUB+GRPCYL,R4           ; GET NUMBER OF GROUPS/CYL
    56 006764  103204  177400                          BIC     #HIBYTE,R4              ; CLEAR UNUSED BITS
    57 006766  107047                          4$:     SUB     R4,R0                   ; SUBTRACT NUMBER OF GROUPS/CYL
    58 006767                                          BMI     5$
       006767  070000  007004                          ^O70000,5$
    59 006771                                          BEQ     6$
       006771  010000  007011                          ^O10000,6$
    60 006773  107200  000001  003567                  SUB     #1,ROFDC                ; DECREMENT STARTING READ ONLY CYL BY 1
    61 006776                                          BCC     4$                      ; IF NO CARRY, LOOP
       006776  040000  006766                          ^O40000,4$
    62 007000  117400  003570                          DEC     ROFDC+1                 ; PROPOGATE BORROW
    63 007002                                          BR      4$                      ; BRANCH
       007002  000000  006766                          ^O0,4$
    64 007004  104203  177777                  5$:     MOV     #-1,R3                  ; SET UP FOR COMPLEMENT
    65 007006  103073                                  BIC     R0,R3                   ; COMPLEMENT (1'S)
    66 007007  115403                                  INC     R3                      ; 2'S COMPLEMENT
    67 007010  104037                                  MOV     R3,R0                   ; RESTORE TO R0
    68 007011  104070  003571                  6$:     MOV     R0,ROFDC+2              ; SAVE IN STARTING GROUP
    69 007013  114003                                  CLR     R3                      ; CLEAR LO ORDER FIRST READ ONLY DBN
    70 007014  114004                                  CLR     R4                      ; CLEAR HI ORDER FIRST READ ONLY DBN
    71 007015  105303  003616                  7$:     ADD     SECGRP,R3               ; ADD SEC/GRP TO LO ORDER STRT SEC
    72 007017                                          BCC     8$                      ; IF NO CARRY, BRANCH
       007017  040000  007022                          ^O40000,8$
    73 007021  115404                                  INC     R4                      ; PROPOGATE CARRY
    74 007022  117405                          8$:     DEC     R5                      ; DECREMENT COUNT
    75 007023                                          BNE     7$                      ; LOOP
       007023  050000  007015                          ^O50000,7$
    76 007025  104030  003565                          MOV     R3,ROFDBN
    77 007027  104040  003566                          MOV     R4,ROFDBN+1
    78 007031  104030  003614                          MOV     R3,CURBLK
    79 007033  104040  003615                          MOV     R4,CURBLK+1
    80 007035  104200  000104  003625                  MOV     #'D,LETTER              ; MAKE DBN'S
```

```
                                        ; POINT TO CURBLK, CALL SOME SUB THAT LOOKS AT THE POINTER PASSED
                                        ; AND THE 'LETTER' AND COMPUTS CYL, TRK, AND GRP. THEN RETURN
                                        ; THEN SEE IF CYL COMPUTED IS > MAX CYL IF SO, THIS PHASE OF TEST IS OVER (BR TGX)
 84 007040  104205  003614              T6G:    MOV     #CURBLK,R5              ;
 85 007042                                      CALL    FNDCY2                  ; GO DIRECTLY TO DBN WILL BE PROCESSED
    007042  020000  002427                      ^O20000,FNDCY2
 86                                     ; THEN SEE IF CYL COMPUTED IS > MAX CYL. IF SO, THIS PHASE OF TEST IS OVER (BR TGX)
 87 007044  106300  003575  003607      1$:     CMP     LDIACYL+1,TSTCYL+1
 88 007047                                      BCS     T6F
    007047  040000  007053                      ^O40000,64$
    007051  000000  007111                      ^O0,T6F
 89 007053  106300  003574  003606              CMP     LDIACYL,TSTCYL
 90 007056                                      BCS     T6F
    007056  040000  007062                      ^O40000,65$
    007060  000000  007111                      ^O0,T6F
 91 007062  104201  003606                      MOV     #TSTCYL,R1
 92 007064                                      CALL    SEEK                    ; SEEK TO REQUESTED CYL
    007064  020000  002621                      ^O20000,SEEK
 93 007066  104205  003614                      MOV     #CURBLK,R5
 94 007070  104303  003603                      MOV     SECTRK,R3
 95 007072                              T6E:    CALL    READ1                   ;READ EACH SECTOR TILL ONE READS OK
    007072  020000  003013                      ^O20000,READ1
 96 007074  103200  177400  003615              BIC     #HIBYTE,CURBLK+1
 97 007077  105300  003603  003614              ADD     SECTRK,CURBLK
 98 007102                                      BCC     T6G
    007102  040000  007040                      ^O40000,T6G
 99 007104  115400  003615                      INC     CURBLK+1
100 007106  115404                             INC     R4
101 007107                                      BR      T6G
    007107  000000  007040                      ^O0,T6G
102
103 007111                              T6F:
```

```
                                        .SBTTL  SEEK TO SELECTED GROUP, READ HDR, SEEK TO DIAGNOSTIC AREA
                                ;STARTING WITH CYLINDER 0, GROUP 0 AND INCREMENTING THROUGH EVERY GROUP
                                ;ON THE DISK, PERFORM A SEEK TO THE SELECTED GROUP, READ A HEADER
                                ;ON THAT GROUP AND THEN A SEEK TO THE FACTORY FORMATTED DIAGNOSTIC
                                ;GROUP TO VERIFY HEADS POSITIONED CORRECTLY.

                                ;COMPUTE LBNS PER GROUP

007111  114001                  T7:     CLR     R1                      ; LBN'S/TRK X TRK/GROUP
007112  104305  003521                  MOV     SUB+TRKGRP,R5
007114  103205  177400                  BIC     #HIBYTE,R5
007116  104303  003527                  MOV     SUB+LBNTRK,R3           ;GET LBN'S PER TRACK
007120  103203  177400                  BIC     #HIBYTE,R3              ;CLEAR UNUSED BITS
007122  104030  003611                  MOV     R3,BLOCKT
007124  105031                  T7A:    ADD     R3,R1
007125  117405                          DEC     R5
007126                                  BNE     T7A
007126  050000  007124                  ^O50000,T7A
007130  104010  003612                  MOV     R1,BLOCKG               ;SAVE IN BLOCKG
007132  114000  003613                  CLR     BLOCKC
007134  104307  003520                  MOV     SUB+GRPCYL,R0
007136  103207  177400                  BIC     #HIBYTE,R0
007140  105300  003612  003613  1$:     ADD     BLOCKG,BLOCKC
007143  117407                          DEC     R0
007144                                  BNE     1$
007144  050000  007140                  ^O50000,1$
007146  114000  003606                  CLR     TSTCYL
007150  114000  003610                  CLR     TSTCYL+2                ;GROUP ZERO
007152  114000  003604                  CLR     TSTBLK
007154  114000  003605                  CLR     TSTBLK+1
007156  114000  003631                  CLR     GRPCNT
007160  114000  003632                  CLR     OLDGRP                  ; CLEAR OLD GROUP VALUE USED IN FNDCYL
007162  104200  000114  003624          MOV     #'L,AREA                ;AREA BEING SEEKED INTO IS LBNS
007165  104300  003624  003625  T7C:    MOV     AREA,LETTER             ; MOVE TO LETTER FOR REPORTING
007170  104205  003604                  MOV     #TSTBLK,R5              ;POINT TO LBN NUMBER FOR -> VAR1
                                ; COMPUTE CYL, TRK AND GRP.
                                ; IF CYL < STARTING CYL THEN CONTINUE ELSE IF CYL >= STARTING DIAG CYL THEN OUT (BR T7X)
                                ; ELSE IF CYL = STARTING XBN CYL CLR TSTBLK, TSTBLK+1 AND BR T7C
007172                          CALL    FNDCYL                          ;FIND CYLINDER
007172  020000  002420                  ^O20000,FNDCYL
007174  106300  003601  003641          CMP     FXBNCYL,CYLLO           ;IS CYL = STARTING XBN?
007177                                  BNE     1$                      ;IF NOT, BRANCH
007177  050000  007237                  ^O50000,1$
007201  106300  003602  003642          CMP     FXBNCYL+1,CYLLO+1       ;
007204                                  BNE     1$
007204  050000  007237                  ^O50000,1$
007206  106200  000130  003624          CMP     #'X,AREA                ; IF HERE AGAIN, BRANCH
007211                                  BEQ     1$
007211  010000  007237                  ^O10000,1$
007213  104200  000130  003624          MOV     #'X,AREA                ; NOW IN XBN AREA
007216  114000  003604                  CLR     TSTBLK                  ;IF SO, CLEAR TSTBLK
007220  114000  003605                  CLR     TSTBLK+1
007222  104300  003603  003611          MOV     SECTRK,BLOCKT           ;CHANGE BLOCK COUNT PER TRACK
007225  104300  003616  003612          MOV     SECGRP,BLOCKG           ;CHANGE BLOCK COUNT PER GROUP
007230  104300  003617  003613          MOV     SECCYL,BLOCKC           ;CHANGE BLOCK COUNT PER CYL
007233  114000  003632                  CLR     OLDGRP                  ; CLEAR OLD GROUP
007235                                  BR      T7C                     ; AND TRY AGAIN WITH XBN
```

```
                007235  000000  007165                          ^O0,T7C
    52          007237  106300  003577  003606  1$:     CMP     FDIACYL,TSTCYL          ;DONE?
    53          007242                                  BNE     2$                      ;IF NOT, CONTINUE
                007242  050000  007251                  ^O50000,2$
    54          007244  106300  003600  003607          CMP     FDIACYL+1,TSTCYL+1      ;DONE?
    55          007247                                  BEQ     T7X                     ;IF SO, EXIT
                007247  010000  007355                  ^O10000,T7X
    56          007251  104300  003624  003625  2$:     MOV     AREA,LETTER
    57          007254  104201  003606                  MOV     #TSTCYL,R1              ;SEEK TO CYLINDER
    58          007256                                  CALL    SEEK
                007256  020000  002621                  ^O20000,SEEK
    59          007260  104205  003604                  MOV     #TSTBLK,R5              ; SET UP POINTER TO TEST BLOCK
    60          007262  104303  003603                  MOV     SECTRK,R3               ;GET MAXIMUM NUMBER OF SECTORS TO READ
    61          007264                                  CALL    READ1                   ;READ UNTIL AT LEAST ONE RECORD READ WITHOUT ERROR
                007264  020000  003013                  ^O20000,READ1
    62                                          ;                       COMPUTE CYL FROM LCDBN
    63          007266  104200  000104  003625  3$:     MOV     #'D,LETTER              ; IN DIAGNOSTIC AREA
    64          007271                                  CALL    FNDCYL
                007271  020000  002420                  ^O20000,FNDCYL
    65          007273  104201  003606                  MOV     #TSTCYL,R1
    66          007275  104303  003603                  MOV     SECTRK,R3               ; R3 = SECTORS/TRACK
    67          007277                                  CALL    SEEK
                007277  020000  002621                  ^O20000,SEEK
    68          007301  104205  003572                  MOV     #LCDBN,R5               ; SET UP POINTER TO TEST BLOCK
    69          007303                                  CALL    READ1                   ;READ UNTIL AT LEAST ONE RECORD READ WITHOUT ERROR
                007303  020000  003013                  ^O20000,READ1
    70          007305  115400  003631                  INC     GRPCNT
    71          007307  103200  177400  003631          BIC     #HIBYTE,GRPCNT
    72          007312                                  BNE     5$
                007312  050000  007335                  ^O50000,5$
    73          007314                                  PUSH    R0                      ; SAVE R0
                007314  100467                                                          MOV R0,-(SP)
    74          007315  104207  060007                  MOV     #T4SOFT,R0              ; TO INTERRUPT HOST <<ONLY>>
    75          007317  104300  002212  001521          MOV     LUNIT,OUT.01            ; IDENTIFY UNIT
    76          007322  114000  001522                  CLR     OUT.02                  ; NO SOFT ERRORS
    77          007324  114000  001523                  CLR     OUT.03                  ; NO ECC CORRECTIONS
    78          007326  114000  001524                  CLR     OUT.04
    79          007330  114000  001525                  CLR     OUT.05
    80          007332                                  CALL    HOSTRQ                  ; REPORT (NOTHING)
                007332  020000  001463                  ^O20000,HOSTRQ
    81          007334                                  POP     R0                      ; RESTORE R0
                007334  104267                                                          MOV (SP)+,R0
    82          007335  105300  003613  003604  5$:     ADD     BLOCKC,TSTBLK
    83          007340                                  BCC     6$
                007340  040000  007344                  ^O40000,6$
    84          007342  115400  003605                  INC     TSTBLK+1
    85          007344  105300  003612  003604  6$:     ADD     BLOCKG,TSTBLK
    86          007347                                  BCC     7$
                007347  040000  007353                  ^O40000,7$
    87          007351  115400  003605                  INC     TSTBLK+1
    88          007353                          7$:     BR      T7C
                007353  000000  007165                  ^O0,T7C
```

```
      1 007355                          T7X:
```

CHECK WRITE PROTECT

```
                                                .SBTTL  CHECK WRITE PROTECT
     5 007355  104200  177777  003627          MOV     #-1,WFLAG               ; SET FLAG
     6 007360  114000  003614                  CLR     CURBLK                  ; CLEAR BLOCK
     7 007362  114000  003615                  CLR     CURBLK+1                ;
     8 007364  114000  003621                  CLR     CURTRK                  ; CLEAR TRACK
     9 007366  104300  C03577  003606          MOV     FDIACYL,TSTCYL          ; POINT TO CYL TO TEST
    10 007371  104300  003600  003607          MOV     FDIACYL+1,TSTCYL+1
    11 007374  114000  003610                  CLR     TSTCYL+2                ; CLEAR GROUP
    12 007376  104201  003606                  MOV     #TSTCYL,R1              ; R1 -> TEST CYL
    13 007400                                  CALL    SEEK                    ; SEEK THERE
       007400  020000  002621                  ^O020000,SEEK
    14 007402                                  CALL    WRITEB                  ; AND ATTEMPT TO WRITE
       007402  020000  011314                  ^O020000,WRITEB
    15 007404  115007                          TST     R0                      ; WAS IT SUCCESSFUL?
    16 007405                                  BNE     1$                      ; IF NOT, THAT WAS EXPECTED
       007405  050000  007434                  ^O050000,1$
    17 007407  115001                          TST     R1                      ;
    18 007410                                  BNE     1$                      ;
       007410  050000  007434                  ^O050000,1$
    19 007412                                  HARDER  35                      ; REPORT ERROR
       007412  104200  002245  001524                          MOV     #MS35,OUT.04
       007415  104300  002212  001523                          MOV     LUNIT,OUT.03
       007420  104202  105733                                  MOV     #35!ERHARD+3000.,R2
       007422  104020  001522                                  MOV     R2,OUT.02
       007424  104200  007424  001521                          MOV     #.,OUT.01
       007427  104200  060013  001520                          MOV     #ERRMES,OUT.R0
    20 007432                                  CALL    TESTEV
       007432  020000  002364                  ^O020000,TESTEV
    21 007434                          1$:     CALL    DR.CLR                  ; CLEAR DRIVE
       007434  020000  002407                  ^O020000,DR.CLR
    22 007436  114000  003627                  CLR     WFLAG                   ; CLEAR FLAG
```

```
                                                .SBTTL  CHANGE MODE TO ALLOW FORMATTING AND WRITING
007440                                  T8:
                                        ;
                                        ;       FIRST CHANGE MODE TO ALLOW FORMATTING AND WRITING
                                        ;
007440  104204  000004                          MOV     #4,R4                   ; R4 HAS DECREMENT COUNTER
007442  104203  003416          2$:     MOV     #CR.MOD,R3              ; POINT TO CHANGE MODE COMMAND
007444  104300  003467  003465          MOV     MODE2,MODE              ; WRITE PROTECT, DIAG CYL ACCESS, NO FORMATTING
007447  104200  002676  003357          MOV     #MS.MOD,COMND           ; SET UP FOR ERROR
007452                                  CALL    TALK                    ; INITIATE SDI INTERCHANGE
007452  020000  001563                  ^O020000,TALK
007454  115002                          TST     R2                      ; SEE IF ERROR OCCURRED
007455                                  BEQ     1$                      ; IF NOT, BRANCH
007455  010000  007466                  ^O010000,1$
007457                                  CALL    TESTEX                  ; IF SO, ERROR
007457  020000  002372                  ^O020000,TESTEX
007461  117404                          DEC     R4                      ; DONE?
007462                                  BNE     2$                      ; IF NOT, BRANCH (TRY AGAIN BECAUSE FORMAT
007462  050000  007442                  ^O050000,2$
                                        ;               WON'T WORK WITH A WRITE PROTECTED DRIVE)
007464                                  BR      TESTX                   ; ELSE, EXIT
007464  000000  002336                  ^O0,TESTX
007466  114000  003630          1$:     CLR     FLAG                    ; USE FLAG TO DECIDE IF WE FORMAT OR TEST A TRACK
007470                                  T08:
                                                .SBTTL  FORMAT A TRACK THEN CHECK IT.
                                        ;
                                        ;       FORMAT ALL TRACKS, THEN DO SEVERAL WRITES, READS AND DATA COMPARES
                                        ;       UNTIL ONE SECTOR PASSES.  DO THIS FOR EACH TRACK IN EACH GROUP
                                        ;       ON EACH CYLINDER IN THE DIAGNOSTIC AREA EXCEPT FOR THE LAST
                                        ;       (FACTORY FORMATTED) CYLINDER ON THE PACK
                                        ;
007470  104300  003577  003606          MOV     FDIACYL,TSTCYL          ; START AT FIRST DIAGNOSTIC CYL
007473  104300  003600  003607          MOV     FDIACYL+1,TSTCYL+1
007476  104201  003606                  MOV     #TSTCYL,R1              ; POINT TO CYLINDER TO SEEK TO
007500  114000  003610                  CLR     TSTCYL+2                ; START WITH GROUP 0
007502                                  CALL    SEEK                    ; SEEK TO FIRST DIAGNOSTIC CYLINDER
007502  020000  002621                  ^O020000,SEEK
007504  114004                          CLR     R4                      ; START WITH DBN 0
007505  104040  003604                  MOV     R4,TSTBLK
007507  104040  003605                  MOV     R4,TSTBLK+1             ; CLEAR BLOCK
007511  104040  003621          T8A:    MOV     R4,CURTRK               ; SAVE TRACK NUMBER
007513  115000  003630                  TST     FLAG                    ; DO WE FORMAT
007515                                  BNE     1$                      ; IF NOT ZERO, TEST TRACK
007515  050000  007523                  ^O050000,1$
007517                                  CALL    FORTRK                  ; FORMAT THE TRACK
007517  020000  010723                  ^O020000,FORTRK
007521                                  BR      2$                      ; DO NOT TEST TRACK UNTIL ALL FORMATTED
007521  000000  007525                  ^O00,2$
007523                          1$:     CALL    TRKTST                  ; TEST THE TRACK
007523  020000  011162                  ^O020000,TRKTST
007525  105300  003603  003604  2$:     ADD     SECTRK,TSTBLK           ; ADD SECTORS/TRACK
007530                                  BCC     T8B                     ; IF NO CARRY, BRANCH
007530  040000  007534                  ^O040000,T8B
007532  115400  003605                  INC     TSTBLK+1                ; INCREMENT
007534  104304  003621          T8B:    MOV     CURTRK,R4               ; GET TRACK NUMBER
007536  115404                          INC     R4                      ; INCREMENT TRACK NUMBER
007537  104303  003521                  MOV     SUB+TRKGRP,R3           ; GET NUMBER OF TRACKS/GROUP
```

FORMAT A TRACK THEN CHECK IT.

```
                                                BIC     #HIBYTE,R3              ; CLEAR UNUSED BITS
                                                CMP     R3,R4                   ; SEE IF ALL TRACKS READ
                                                BNE     T8A                     ; IF NOT, BRANCH
    007544  050000  007511                      ^O50000,T8A
                                                CLR     R4                      ; ZERO R4
                                                INC     TSTCYL+2                ; MOVE TO NEXT GROUP
                                                MOV     SUB+GRPCYL,R1           ; GET GROUPS/CYLINDER
                                                BIC     #HIBYTE,R1              ; CLEAR UNUSED BITS
                                                CMP     R1,TSTCYL+2             ; COMPARE
                                                BNE     T8D                     ; IF ALL GROUPS NOT TESTED, BRANCH
    007557  050000  007571                      ^O50000,T8D
                                                MOV     R4,TSTCYL+2             ; START WITH GROUP 0 ON NEW CYLINDER
                                                INC     TSTCYL                  ; INCREMENT CYLINDER
                                                BCC     T8D                     ; IF NO CARRY, BRANCH
    007565  040000  007571                      ^O40000,T8D
                                                INC     TSTCYL+1                ; INCREMENT HI CYLINDER
                                        T8D:    CMP     TSTCYL,ROFDC            ; SEE IF ON LAST CYLINDER (LO ORDER)
                                                BNE     1$                      ; IF NOT, BRANCH
    007574  050000  007610                      ^O50000,1$
                                                CMP     TSTCYL+1,ROFDC+1        ; SEE IF ON LAST CYLINDER (HI ORDER)
                                                BNE     1$                      ; IF NOT, BRANCH
    007601  050000  007610                      ^O50000,1$
                                                CMP     TSTCYL+2,ROFDC+2        ; SEE IF ON RESERVED GROUPS
                                                BEQ     T8T                     ; IF SO, END THIS PHASE OF TEST
    007606  010000  007616                      ^O10000,T8T
                                        1$:     MOV     #TSTCYL,R1              ; POINT TO NEW CYLINDER OR GROUP
                                                CALL    SEEK                    ; ISSUE SEEK
    007612  020000  002621                      ^O20000,SEEK
                                                BR      T8A                     ; BRANCH
    007614  000000  007511                      ^O0,T8A
                                        T8T:    TST     FLAG                    ; ARE WE DONE?
                                                BNE     T8E                     ; IF SO, BRANCH
    007620  050000  007627                      ^O50000,T8E
                                                MOV     #-1,FLAG                ; ELSE, CHANGE FLAG
                                                BR      T08                     ; AND CHECK TRACKS
    007625  000000  007470                      ^O0,T08

                                        ; *** NOW SEND INVALID SELECT GROUP COMMAND AND EXPECT ERROR
                                        T8E:    MOV     #1,SDILTO               ; CHANGE LONG TIMEOUT
                                                MOV     #SUB+GRPCYL,R0          ; R0 HAS GROUP #
                                                BIC     #HIBYTE,R0              ; STRIP OFF UNUSED BITS
                                                CMP     #LOBYTE,R0              ; IS THIS UNIT USE MAX BITS?
                                                BEQ     T11                     ; IF SO, BRANCH (DON'T DO TEST)
    007640  010000  007717                      ^O10000,T11
                                                BIS     #SL.GRP,R0              ; SET SELECT GROUP CODE
                                                MOV     R0,RUN                  ; STORE IN RUN CODE
                                                CLR     CR.RUN+1                ; SET UP FOR LEVEL 1 EXCHANGE
                                                MOV     #CR.RUN,R3              ; R3 -> PACKET
                                                MOV     SDI,R2                  ; SEND MESSAGE
                                                MOV     (R3)+,R0                ; SET UP PARAMETERS
                                                MOV     (R3),R1                 ;
                                                XFC     SEND                    ; SEND COMMAND
                                                TST     R1                      ; IF FAIL, BRANCH
                                                BNE     T8F
    007660  050000  007715                      ^O50000,T8F
                                                MOV     #CR.GST,R3              ; CHECK OUT STATUS
                                                CALL    TALK
```

FORMAT A TRACK THEN CHECK IT.

```
        007664  020000  001563                  ^O020000,TALK
     92 007666  102200  000350  003476          BIT     #<ST.PE+ST.RE+ST.FE+ST.WE>,ST.ERR+ST ;ERROR?
     93 007671                                  BNE     T8F                     ; IF SO, BRANCH
        007671  050000  007715                  ^O050000,T8F
     94 007673                                  HARDER  33                      ; ELSE ERROR
        007673  104200  002156  001524                                                  MOV     #MS33,OUT.04
        007676  104300  002212  001523                                                  MOV     LUNIT,OUT.03
        007701  104202  105731                                                          MOV     #33!ERHARD+3000.,R2
        007703  104020  001522                                                          MOV     R2,OUT.02
        007705  104200  007705  001521                                                  MOV     #.,OUT.01
        007710  104200  060013  001520                                                  MOV     #ERRMES,OUT.RQ
     95 007713                                  CALL    TESTEV
        007713  020000  002364                  ^O020000,TESTEV
     96 007715                          T8F:    CALL    DR.CLR                  ; CLEAR DRIVE OF ERROR
        007715  020000  002407                  ^O020000,DR.CLR
```

```
                                                .SBTTL  SEND INVALID LEVEL 2 COMMANDS
007717  114000  003357          T11:    CLR     COMND                   ; CLEAR COMMAND CODE AS A FLAG
007721  114007                          CLR     R0                      ; CLEAR SDI COMMAND VALUE
007722  104200  000001  003436          MOV     #1,CR.RUN+1             ; RESET PROPER LENGTH
007725  104070  003456          1$:     MOV     R0,RUN                  ; SET CODE
007727                                  PUSH    R0                      ; SAVE R0
        007727  100467                                                          MOV R0,-(SP)
007730                                  CALL    TSTCMD                  ; AND TEST
        007730  020000  010566          ^020000,TSTCMD
007732                                  POP     R0                      ; RESTORE R0
        007732  104267                                                          MOV (SP)+,R0
007733  115007                          TST     R0                      ; TRY INVALID OPCODE ONCE
007734                                  BNE     2$                      ; EXIT WHEN TRIED ONCE WITH INVALID CODE
        007734  050000  007742          ^050000,2$
007736  105207  000400                  ADD     #400,R0                 ; ADD TO NEXT CODE
007740                                  BNE     1$                      ; IF NOT DONE, TEST AGAIN
        007740  050000  007725          ^050000,1$
007742  104200  000377  003456  2$:     MOV     #377,RUN                ; TRY WITH LOW BITS SET
007745                                  CALL    TSTCMD                  ;
        007745  020000  010566          ^020000,TSTCMD
007747  104300  003470  003456          MOV     RUNCMD,RUN              ; RESTORE RUN COMMAND
007752  104205  000100                  MOV     #64.,R5
007754  104050  003436                  MOV     R5,CR.RUN+1             ; STORE INVALID COMMAND LENGTH
007756  104204  003415                  MOV     #SER50,R4               ;
007760                                  CALL    TSTCMD
        007760  020000  010566          ^020000,TSTCMD
007762  114005                          CLR     R5                      ; TRY AGAIN WITH 0 COMMAND LENGTH BUT INVALID COMMA'
007763  104204  003422                  MOV     #SER51,R4
007765  104050  003414                  MOV     R5,CR.GST+3             ; DESTROY RESPONSE LENGTH
007767  104203  0034'                   MOV     #CR.GST,R3
007771                                  CALL    TSTCM2
        007771  020000  010570          ^020000,TSTCM2
007773  104200  000007  003414          MOV     #7,CR.GST+3             ; RESTORE COMMAND LENGTH
```

SEND INVALID LEVEL 1 COMMANDS

```
                                        .SBTTL  SEND INVALID LEVEL 1 COMMANDS

                                ; *** SEND START THEN SELECT GROUP
007776  114000  003436                   CLR     CR.RUN+1
010000  104200  070400  003456           MOV     #MS.STR,RUN             ; SET UP COMMAND
010003                                   CALL    SNDLV1                  ; SEND LEVEL 1 START COMMAND
010003  020000  010552                   ^O20000,SNDLV1
010005  104200  003430  003357           MOV     #SER52,COMND
010010  104200  107000  003456           MOV     #SL.GRP,RUN             ; MOVE SELECT GROUP CODE INTO COMMAND
010013                                   CALL    SNDL1A                  ; SEND NEXT FRAME
010013  020000  010555                   ^O20000,SNDL1A
010015  104200  131000  003456           MOV     #MS.END,RUN             ; SET UP END FRAME
010020                                   CALL    TSTCMD                  ; CHECK IF CAUGHT
010020  020000  010566                   ^O20000,TSTCMD

                                ; *** SEND END WITH NO START
010022  104200  131000  003456           MOV     #MS.END,RUN             ; SET UP END FRAME
010025  104200  003467  003357           MOV     #SER53,COMND
010030                                   CALL    TSTCMD                  ; CHECK IF CAUGHT
010030  020000  010566                   ^O20000,TSTCMD

                                ; *** SEND START THEN END WITH BAD CHECKSUM
010032  104200  070400  003456           MOV     #MS.STR,RUN             ; SET UP START FRAME
010035                                   CALL    SNDLV1                  ; SEND FRAME
010035  020000  010552                   ^O20000,SNDLV1
010037  104200  003520  003357           MOV     #SER54,COMND
010042  104200  131000  003456           MOV     #MS.END,RUN             ; SET UP END FRAME (ZERO CHECKSUM)
010045                                   CALL    TSTCMD                  ; CHECK IF CAUGHT
010045  020000  010566                   ^O20000,TSTCMD

                                ; *** SEND CONTINUE WITH NO START
010047  104200  152000  003456           MOV     #MS.CNT,RUN             ; SET CONTINUE FRAME
010052  101200  000014  003456           BIS     #RUNLBC,RUN
010055                                   CALL    SNDLV1                  ; SEND IT
010055  020000  010552                   ^O20000,SNDLV1
010057  104200  131000  003456           MOV     #MS.END,RUN             ; SET UP END FRAME
010062  104200  003551  003357           MOV     #SER55,COMND
010065                                   CALL    TSTCMD                  ; CHECK IF CAUGHT
010065  020000  010566                   ^O20000,TSTCMD
```

```
                                        ;
                                        ;       FIND IF DRIVE WAS FORMATTED IN 512 BYTE MODE.
                                        ;
 010067  104307  003471         T5:    MOV     COPY,R0                 ; GET NUMBER OF XBN COPIES
 010071  105077                         ADD     R0,R0                   ; DOUBLE
 010072  104070  003473                 MOV     R0,SERRTY               ; RETRY 2 TIMES THE NUMBER OF COPIES
 010074  114000  003472         1$:     CLR     RETRY                   ; TO SET UP COPIES AND SETUP CALC
 010076  114000  003614                 CLR     CURBLK                  ; PUT LO ORDER XBN IN CURBLK
 010100  114000  003615                 CLR     CURBLK+1                        ; PUT HI ORDER XBN IN CURBLK+1
 010102  104205  003614         2$:     MOV     #CURBLK,R5
 010104  104204  003577                 MOV     #FDIACYL,R4
 010106  104303  003603                 MOV     SECTRK,R3
 010110  114001                         CLR     R1
 010111                                 CALL    FNDCYL
 010111  020000  002420                 ^O20000,FNDCYL
 010113  104302  003356                 MOV     SDI,R2
 010115  104203  003430                 MOV     #CR.INR,R3              ; SET UP FOR RECALIBRATE
 010117                                 CALL    TALK                    ; RECALIBRATE DRIVE
 010117  020000  001563                 ^O20000,TALK
 010121  115002                         TST     R2                      ; SEE IF ERROR OCCURRED
 010122                                 BNE     8$                      ; IF SO, BRANCH
 010122  050000  010277                 ^O50000,8$
 010124  104201  003641                 MOV     #CYLLO,R1
 010126                                 CALL    SEEK
 010126  020000  002621                 ^O20000,SEEK
 010130  104301  002174         3$:     MOV     SDILTO,R1               ; INNER LOOP TIMEOUT
 010132  114005                         CLR     R5
 010133                         4$:     CALL    RTDSL                   ; GET REAL TIME DRIVE STATE
 010133  020000  001422                 ^O20000,RTDSL
 010135  115002                         TST     R2                      ; SEE IF ERROR OCCURRED
 010136                                 BNE     8$                      ; IF SO, BRANCH
 010136  050000  010277                 ^O50000,8$
 010140  115001                         TST     R1                      ; SEE IF READ/WRITE READY IS ASSERTED
 010141                                 BMI     5$                      ; IF SO, BRANCH
 010141  070000  010153                 ^O70000,5$
 010143                                 ASSUME  RWRDY,100000
 010143  117405                         DEC     R5
 010144                                 BNE     4$
 010144  050000  010133                 ^O50000,4$
 010146  117404                         DEC     R4
 010147                                 BNE     4$
 010147  050000  010133                 ^O50000,4$
 010151                                 BR      8$                      ; ERROR IF TIMED OUT
 010151  000000  010277                 ^O0,8$
 010153  104200  100000  003647  5$:    MOV     #RSTOP,CHAIN+RW.CPT     ; MOVE LAST CHAIN FLAG TO CHAIN
 010156  104300  003614  003652         MOV     CURBLK,CHAIN+RW.LOW     ; MOVE TO OUTPUT
 010161  104301  003520                 MOV     SUB+HiXBN,R1            ; GET HI XBN BITS
 010163  110601                         ROR     R1                      ; ROTATE TO CORRECT POSITION
 010164  110601                         ROR     R1                      ; ROTATE TO CORRECT POSITION
 010165  110601                         ROR     R1                      ; ROTATE TO CORRECT POSITION
 010166  110601                         ROR     R1                      ; ROTATE TO CORRECT POSITION
 010167  103201  170377                 BIC     #HBLONB,R1              ; CLEAR UNUSED BITS
 010171  101201  120000                 BIS     #HD.XBN,R1              ; MAKE A XBN
 010173  105301  003615                 ADD     CURBLK+1,R1             ; SET IN HI XBN BITS
 010175  104010  003653                 MOV     R1,CHAIN+RW.HI          ; SAVE
 010177  104300  003644  003654         MOV     TRACK,CHAIN+RW.CMD      ; MOVE TRACK TO CHAIN
 010202  101200  013400  003654         BIS     #RREAL,CHAIN+RW.CMD     ; SET IN REAL TIME COMMAND
```

```
     59 010205  104207  004377                    MOV     #RBUFD,R0
     60 010207  104070  003651                    MOV     R0,CHAIN+RW.BUF
     61 010211  104302  003356                    MOV     SDI,R2                  ; R2 HAS UDA PORT MASK
     62 010213  104207  003647                    MOV     #CHAIN,R0               ; R0 POINTS TO CHAIN
     63 010215  060012                            XFC     WAITSI                  ; WAIT FOR SECTOR OR INDEX
     64 010216  115001                            TST     R1                      ; SEE IF ERROR OCCURRED
     65 010217                                    BNE     8$                      ; IF SO, BRANCH
        010217  050000  010277                    ^050000,8$
     69          000400              SEC512 = 256.
     70          000440              SEC576 = 288.
     71 010221  104202  000400                    MOV     #SEC512,R2              ; READ A 512 BYTE SECTOR
     75 010223  060002                            XFC     XREAD                   ; READ THE SECTOR
     76 010224  106201  000004                    CMP     #4,R1                   ; SEE IF XBN HEADER COMPARE FAILURE
     77 010226                                    BEQ     7$                      ; IF SO, TRY NEXT COPY
        010226  010000  010257                    ^010000,7$
     78 010230  115001                            TST     R1                      ; SEE IF AN ERROR OCCURRED
     79 010231                                    BNE     8$                      ; IF SO, BRANCH
        010231  050000  010277                    ^050000,8$
     88 010233  102200  010000  003647            BIT     #ECCFLG,CHAIN+RW.CPT    ; SEE IF BUFFER HAS AN ECC ERROR
     92 010236                                    BEQ     6$                      ; IF NOT, BRANCH
        010236  010000  010255                    ^010000,6$
     93 010240  104207  003647                    MOV     #CHAIN,R0               ; POINT TO CHAIN
     94 010242  060015                            XFC     ECC                     ; CORRECT THE BUFFER
     95 010243  115001                            TST     R1                      ; SEE IF ERROR
     96 010244                                    BNE     7$                      ; IF SO, BRANCH
        010244  050000  010257                    ^050000,7$
     97 010246  104307  003476                    MOV     ST+ECCRSH,R0
     98 010250  103207  177400                    BIC     #HIBYTE,R0
     99 010252  106071                            CMP     R0,R1
    100 010253                                    BMI     7$
        010253  070000  010257                    ^070000,7$
    101 010255                          6$:       BR      9$                      ;
        010255  000000  010330                    ^00,9$
    102 010257  115400  003472          7$:       INC     RETRY                   ; INCREMENT CURRENT COPY NUMBER
    103 010261  106300  003471  003472            CMP     COPY,RETRY              ; SEE IF ALL COPIES TRIED
    104 010264                                    BEQ     8$                      ; IF SO, BRANCH
        010264  010000  010277                    ^010000,8$
    105 010266  105300  003513  003614            ADD     CR+FCTSIZ,CURBLK        ; ADD TO CURRENT BLOCK
    106 010271                                    BCC     2$                      ; CALCULATE AND TRY NEXT SECTOR
        010271  040000  010102                    ^040000,2$
    107 010273  115400  003615                    INC     CURBLK+1                        ; PROPOGATE CARRY
    108 010275                                    BR      2$                      ; BRANCH
        010275  000000  010102                    ^00,2$
    109 010277  117400  003473          8$:       DEC     SERRTY                  ; DECREMENT RETRY COUNT
    110 010301                                    BNE     1$                      ; IF UNEXHAUSTED, BRANCH
        010301  050000  010074                    ^050000,1$
    111 010303                                    HARDER  38,#SER36
        010303  104200  002422  001524                                                    MOV     #MS38,OUT.04
        010306  104200  002446  001525                                                    MOV     #SER36,OUT.05
        010311  104300  002212  001523                                                    MOV     LUNIT,OUT.03
        010314  104202  105736                                                            MOV     #38!ERHARD+3000.,R2
        010316  104020  001522                                                            MOV     R2,OUT.02
        010320  104200  010320  001521                                                    MOV     #.,OUT.01
        010323  104200  060013  001520                                                    MOV     #ERRMES,OUT.RQ
    112 010326                                    BR      11$
        010326  000000  010407                    ^00,11$
    113 010330  106200  126736  004377  9$:       CMP     #MOD512,RBUFD           ; SEE IF 512 BYTE MODE DRIVE
```

```
    114 010333                                          BEQ     10$                     ; IF SO, BRANCH
        010333  010000  010411                          ^010000,10$
    115 010335  106200  074161  004377                  CMP     #MOD576,RBUFD           ; SEE IF 576 BYTE MODE DRIVE
    116 010340                                          BEQ     12$
        010340  010000  010367                          ^010000,12$
    117 010342                                          HARDER  36,#SER36               ; SET UP REPORT
        010342  104200  002306  001524                                                      MOV     #MS36,OUT.04
        010345  104200  002446  001525                                                      MOV     #SER36,OUT.05
        010350  104300  002212  001523                                                      MOV     LUNIT,OUT.03
        010353  104202  105734                                                              MOV     #36!ERHARD+3000.,R2
        010355  104020  001522                                                              MOV     R2,OUT.02
        010357  104200  010357  001521                                                      MOV     #.,OUT.01
        010362  104200  060013  001520                                                      MOV     #ERRMES,OUT.RQ
    118 010365                                          BR      11$
        010365  000000  010407                          ^00,11$
    119 010367                                  12$:    HARDER  37
        010367  104200  002331  001524                                                      MOV     #MS37,OUT.04
        010372  104300  002212  001523                                                      MOV     LUNIT,OUT.03
        010375  104202  105735                                                              MOV     #37!ERHARD+3000.,R2
        010377  104020  001522                                                              MOV     R2,OUT.02
        010401  104200  010401  001521                                                      MOV     #.,OUT.01
        010404  104200  060013  001520                                                      MOV     #ERRMES,OUT.RQ
    120 010407                                  11$:    CALL    TESTED
        010407  020000  002400                          ^020000,TESTED
    121 010411                                  10$:
```

ISSUE DISCONNECT COMMAND

```
      1                                          .SBTTL  ISSUE DISCONNECT COMMAND
      2 010411  104203  003377          T9:     MOV     #CR.DIS,R3              ; POINT TO DISCONNECT COMMAND
      3 010413  104200  002621  003357          MOV     #MS.DIS,COMND           ; SET UP FOR ERROR
      4 010416                                   CALL    TALK                    ; INITIATE SDI INTERCHANGE
        010416  020000  001563                   ^O20000,TALK
      5 010420  115002                           TST     R2                      ; SEE IF ERROR OCCURRED
      6 010421                                   BEQ     T10
        010421  010000  010425                   ^O10000,T10
      7 010423                                   CALL    TESTEX                  ; IF SO, BRANCH
        010423  020000  002372                   ^O20000,TESTEX
      8
      9                                          .SBTTL  CHECK AVAIL FLAG
     10 010425  114005                  T10:    CLR     R5                      ; SET UP TIMEOUT COUNTER
     11 010426  104302  003356          T10LOP: MOV     SDI,R2                  ; MOVE MASK TO R2
     12 010430                                   CALL    RDSTAT                  ; GET DRIVE STATUS
        010430  020000  001432                   ^O20000,RDSTAT
     13 010432  115002                           TST     R2                      ; WAS IT OK?
     14 010433                                   BEQ     2$                      ; IF NO ERROR, BRANCH
        010433  010000  010505                   ^O10000,2$
     15 010435  102201  000400                   BIT     #RCVERR,R1              ; SEE IF ANY ERRORS
     16 010437                                   BNE     1$                      ; IF NOT, ERROR
        010437  050000  010463                   ^O50000,1$
     17 010441                                   HARDER  11                      ; REPORT INVALID STATUS ERROR
        010441  104200  000366  001524                           MOV     #MS11,OUT.04
        010444  104300  002212  001523                           MOV     LUNIT,OUT.03
        010447  104202  105703                                   MOV     #11!ERHARD+3000.,R2
        010451  104020  001522                                   MOV     R2,OUT.02
        010453  104200  010453  001521                           MOV     #.,OUT.01
        010456  104200  060013  001520                           MOV     #ERRMES,OUT.RQ
     18 010461                                   BR      TESTEW                  ; BRANCH TO DONE
        010461  000000  002355                   ^O0,TESTEW
     19 010463                          1$:
     20 010463                                   HARDER  12                      ; REPORT XMIT ERROR
        010463  104200  000434  001524                           MOV     #MS12,OUT.04
        010466  104300  002212  001523                           MOV     LUNIT,OUT.03
        010471  104202  105704                                   MOV     #12!ERHARD+3000.,R2
        010473  104020  001522                                   MOV     R2,OUT.02
        010475  104200  010475  001521                           MOV     #.,OUT.01
        010500  104200  060013  001520                           MOV     #ERRMES,OUT.RQ
     21 010503                                   BR      TESTEW                  ; BRANCH TO DONE
        010503  000000  002355                   ^O0,TESTEW
     22 010505                          2$:
     23 010505  102201  000100                   BIT     #AVAIL,R1               ; SEE IF AVAILABLE IS ASSERTED
     24 010507                                   BNE     T12                     ; IF SO, BRANCH TO LAST TEST
        010507  050000  010542                   ^O50000,T12
     25 010511  117405                           DEC     R5                      ; DECREMENT TIMEOUT COUNT
     26 010512                                   BNE     T10LOP                  ; IF UNEXPIRED, BRANCH
        010512  050000  010426                   ^O50000,T10LOP
     27 010514  103201  077674                   BIC     #077674,R1              ; CLEAR UNUSED BITS
     28 010516                                   HARDER  29,R1                   ; REPORT ERROR
        010516  104200  002007  001524                           MOV     #MS29,OUT.04
        010521  104010  001525                                   MOV     R1,OUT.05
        010523  104300  002212  001523                           MOV     LUNIT,OUT.03
        010526  104202  105725                                   MOV     #29!ERHARD+3000.,R2
        010530  104020  001522                                   MOV     R2,OUT.02
        010532  104200  010532  001521                           MOV     #.,OUT.01
        010535  104200  060013  001520                           MOV     #ERRMES,OUT.RQ
```

CHECK AVAIL FLAG

```
     29 010540                                      CALL    TESTEV
        010540  020000  002364                      ^O20000,TESTEV
     30
     31
     32 010542                          T12:
     33 010542  104300  003470  003456              MOV     RUNCMD,RUN          ; RESTORE RUN COMMAND
     34 010545  104200  000001  003436              MOV     #1,CR.RUN+1         ; RESTORE COMMAND LENGTH
     35 010550                                      BR      TESTX               ; BRANCH TO TEST NEXT UNIT
        010550  000000  002336                      ^O0,TESTX
```

```
                                        .SBTTL  SNDLV1 AND TSTCMD
                                ; *** SEND LEVEL 1 START COMMAND
                                ;       SEND THE START FRAME AND RETURN
010552  101200  000014  003456  SNDLV1: BIS     #RUNLBC,RUN             ; SET RUN COMMAND
010555  104203  003435          SNDL1A: MOV     #CR.RUN,R3              ; SEND COMMAND
010557  104302  003356                  MOV     SDI,R2                  ; R2 -> PORT
010561  104237                         MOV     (R3)+,R0                ; SET UP PARAMETERS
010562  104131                         MOV     (R3),R1
010563  060004                         XFC     SEND                    ; SEND COMMAND
010564                                 RETURN
010564  000000  000000                 ^O0,0

                                ; *** TEST THE COMMAND
                                ;       INPUT   RUN HAS LEVEL 1 OR 2 COMMAND SET UP
                                ;               CR.RUN = 0 TO INITIATE LEVEL 1 TRANMITTION
                                ;                      = WHATEVER FOR LEVEL 2
                                ;       OUTPUT  WILL NOT RETURN IF ERROR OCCURED
                                ;
010566  104203  003435          TSTCMD: MOV     #CR.RUN,R3              ; SEND COMMAND
010570                          TSTCM2: CALL    TALK
010570  020000  001563                  ^O20000,TALK
010572  115002                         TST     R2                      ; DID IT FAIL?
010573                                 BEQ     T11ER                   ; IF NOT, BRANCH TO REPORT AND EXIT
010573  010000  010606                 ^O10000,T11ER
010575  104203  003411                 MOV     #CR.GST,R3              ; CHECK STATUS
010577                                 CALL    TALK                    ; GET STATUS
010577  020000  001563                 ^O20000,TALK
010601  102200  000350  003476         BIT     #<ST.PE+ST.RE+ST.FE+ST.WE>,ST+ST.ERR
010604                                 BNE     TSTCME                  ; BRANCH IF ERROR
010604  050000  010717                 ^O50000,TSTCME
                                ; *** DID NOT FAIL -> ERROR
010606  115000  003357          T11ER:  TST     COMND                   ; DID WE HAVE COMMAND SET
010610                                 BNE     T11X2                   ; IF SO, LEVEL 1 XMIT ERROR
010610  050000  010672                 ^O50000,T11X2
010612  106300  003470  003456         CMP     RUNCMD,RUN              ; ELSE, WHAT TYPE LEVEL 2 XMIT ERROR?
010615                                 BNE     T11X1                   ; IF RUN NOT SET TO RUN COMMAND, BRANCH
010615  050000  010645                 ^O50000,T11X1
010617                                 HARDER  31,<R4,R5>
010617  104200  002077  001524                 MOV     #MS31,OUT.04
010622  104040  001525                         MOV     R4,OUT.05
010624  104050  001526                         MOV     R5,OUT.06
010626  104300  002212  001523                 MOV     LUNIT,OUT.03
010631  104202  105727                         MOV     #31!ERHARD+3000.,R2
010633  104020  001522                         MOV     R2,OUT.02
010635  104200  010635  001521                 MOV     #.,OUT.01
010640  104200  060013  001520                 MOV     #ERRMES,OUT.RQ
010643                                 BR      TESTED
010643  000000  002400                 ^O0,TESTED
010645                          T11X1:  HARDER  30,RUN
010645  104200  002053  001524                 MOV     #MS30,OUT.04
010650  104300  003456  001525                 MOV     RUN,OUT.05
010653  104300  002212  001523                 MOV     LUNIT,OUT.03
010656  104202  105726                         MOV     #30!ERHARD+3000.,R2
010660  104020  001522                         MOV     R2,OUT.02
010662  104200  010662  001521                 MOV     #.,OUT.01
010665  104200  060013  001520                 MOV     #ERRMES,OUT.RQ
010670                                 BR      TESTED
```

```
      010670  000000  002400                  ^O0,TESTED
   35 010672                          T11X2:  HARDER  32,COMND
      010672  104200  002130  001524                                  MOV     #MS32,OUT.04
      010675  104300  003357  001525                                  MOV     COMND,OUT.05
      010700  104300  002212  001523                                  MOV     LUNIT,OUT.03
      010703  104202  105730                                          MOV     #32!ERHARD+3000.,R2
      010705  104020  001522                                          MOV     R2,OUT.02
      010707  104200  010707  001521                                  MOV     #.,OUT.01
      010712  104200  060013  001520                                  MOV     #ERRMES,OUT.RQ
   36 010715                                  BR      TESTED
      010715  000000  002400                  ^O0,TESTED
   37 010717                          TSTCME: CALL    DR.CLR                  ; CLEAR ERROR
      010717  020000  002407                  ^O20000,DR.CLR
   38 010721                                  RETURN
      010721  000000  000000                  ^O0,0
```

```
                                                .SBTTL  FORTRK - FORMAT TRACK
010723                                  FORTRK:
                                        ;
                                        ;       FORMAT TRACK GIVEN BY CURTRK, STARTING WITH DBN GIVEN IN TSTBLK
                                        ;       THE DATA WRITTEN IS UNPREDICTABLE
                                        ;
010723  104307  003604                          MOV     TSTBLK,R0               ; GET LO STARTINC DBN
010725  104301  003521                          MOV     SUB+HIDBN,R1            ; GET HIGH ORDER BITS OF STARTING DBN
010727  110601                                  ROR     R1                      ; MOVE TO CORRECT POSITION
010730  110601                                  ROR     R1
010731  110601                                  ROR     R1
010732  110601                                  ROR     R1
010733  103201  170377                          BIC     #HBLONB,R1              ; CLEAR UNUSED BITS
010735  101201  140000                          BIS     #HD.DBN,R1              ; SET HEADER CODE
010737  101301  003605                          BIS     TSTBLK+1,R1             ; GET HI STARTING DBN
010741  104203  003767                          MOV     #OBUFF,R3               ; POINT TO OUTPUT BUFFER
010743  104304  003603                          MOV     SECTRK,R4               ; R4 CONTAINS NUMBER OF SECTORS TO FORMAT
010745  104205  004377                          MOV     #FCHAIN,R5              ; R5 POINTS TO FORMAT CHAIN
010747  100253                          FLOOP:  MOV     R3,(R5)+                ; MOVE POINTER TO BUFFER TO CHAIN
010750  100257                                  MOV     R0,(R5)+                ; MOVE LO DBN TO CHAIN
010751  100251                                  MOV     R1,(R5)+                ; MOVE HI DBN TO CHAIN
010752  115407                                  INC     R0                      ; INCREMENT LO DBN
010753                                          BCC     FCARY                   ; IF NO CARRY, BRANCH
010753  040000  010756                          ^O040000,FCARY
010755  115401                                  INC     R1                      ; INCREMENT HI DBN
010756  117404                          FCARY:  DEC     R4                      ; DECREMENT SECTOR COUNT
010757                                          BNE     FLOOP                   ; BRANCH IF COUNT UNEXHAUSTED
010757  050000  010747                          ^O050000,FLOOP
010761  104207  100000                          MOV     #FSTOP,R0               ; GET FORMAT END-OF-CHAIN FLAG
010763  100157                                  MOV     R0,(R5)                 ; MOVE INTO CHAIN
010764  104302  003356                          MOV     SDI,R2                  ; MOVE MASK TO R2
010766                                          CALL    RDSTAT                  ; GET DRIVE STATUS
010766  020000  001432                          ^O020000,RDSTAT
010770  115002                                  TST     R2                      ; WAS IT OK?
010771                                          BEQ     2$                      ; IF NO ERROR, BRANCH
010771  010000  011043                          ^O010000,2$
010773  102201  000400                          BIT     #RCVERR,R1              ; SEE IF ANY ERRORS
010775                                          BNE     1$                      ; IF NOT, ERROR
010775  050000  011021                          ^O050000,1$
010777                                          HARDER  10                      ; REPORT INVALID STATUS ERROR
010777  104200  000341  001524                                                          MOV     #MS10,OUT.04
011002  104300  002212  001523                                                          MOV     LUNIT,OUT.03
011005  104202  105702                                                                  MOV     #10!ERHARD+3000.,R2
011007  104020  001522                                                                  MOV     R2,OUT.02
011011  104200  011011  001521                                                          MOV     #.,OUT.01
011014  104200  060013  001520                                                          MOV     #ERRMES,OUT.RQ
011017                                          BR      TESTEW          ; BRANCH TO DONE
011017  000000  002355                          ^O0,TESTEW
011021                                  1$:
011021                                          HARDER  12                      ; REPORT XMIT ERROR
011021  104200  000434  001524                                                          MOV     #MS12,OUT.04
011024  104300  002212  001523                                                          MOV     LUNIT,OUT.03
011027  104202  105704                                                                  MOV     #12!ERHARD+3000.,R2
011031  104020  001522                                                                  MOV     R2,OUT.02
011033  104200  011033  001521                                                          MOV     #.,OUT.01
011036  104200  060013  001520                                                          MOV     #ERRMES,OUT.RQ
011041                                          BR      TESTEW                  ; BRANCH TO DONE
```

```
     011041  000000  002355                  ^O0,TESTEW
  40 011043                          2$:
  41 011043  102201  100000                  BIT     #RWRDY,R1               ; TEST R/W READY
  42 011045                                  BNE     FGO                     ; IF ASSERTED, BRANCH
     011045  050000  011071                  ^O50000,FGO
  43 011047                                  HARDER  18                      ;READ/WRITE READY DROPPED BEFORE FORMAT
     011047  104200  000737  001524                                                  MOV     #MS18,OUT.04
     011052  104300  002212  001523                                                  MOV     LUNIT,OUT.03
     011055  104202  105712                                                         MOV     #18!ERHARD+3000.,R2
     011057  104020  001522                                                         MOV     R2,OUT.02
     011061  104200  011061  001521                                                 MOV     #.,OUT.01
     011064  104200  060013  001520                                                 MOV     #ERRMES,OUT.RQ
  44 011067                                  BR      TESTEV                  ; BRANCH TO EXIT
     011067  000000  002364                  ^O0,TESTEV
  45 011071  104207  004377          FGO:    MOV     #FCHAIN,R0              ; POINT TO FORMAT CHAIN (FOR XFC)
  46 011073  104304  003523                  MOV     SUB+DATPRE,R4           ; GET DATA PREAMBLE LENGTHS
  47 011075  103204  177400                  BIC     #HIBYTE,R4              ; CLEAR UNUSED BITS
  48 011077  104303  003523                  MOV     SUB+HDRPRE,R3           ; GET HEADER PREAMBLE LENGTH
  49 011101  110703                          SWAB    R3                      ; SWAP BYTES
  50 011102  103203  177400                  BIC     #HIBYTE,R3              ; CLEAR UNUSED BYTES
  51 011104  104301  003621                  MOV     CURTRK,R1               ; TRACK NUMBER
  55 011106  104202  000400                  MOV     #BUFFSZ,R2              ; R2 = BUFFER SIZE FOR 52
  64 011110  060001                          XFC     FORMAT                  ; FORMAT THE TRACK (BUFFER CONTENTS ARE A DON'T CARE
  65 011111  115001                          TST     R1                      ; SEE IF ANY ERRORS OCCURRED
  66 011112                                  BEQ     FOREXT                  ; IF NOT, BRANCH
     011112  010000  011160                  ^O10000,FOREXT
  67 011114                                  HARDER  19,<INS+1,INS+2,INS+3,CURTRK> ;TIMEOUT OF DRIVE OR READ/WRITE READY DROPPED
     011114  104200  000771  001524                                                  MOV     #MS19,OUT.04
     011117  104300  003461  001525                                                  MOV     INS+1,OUT.05
     011122  104300  003462  001526                                                  MOV     INS+2,OUT.06
     011125  104300  003463  001527                                                  MOV     INS+3,OUT.07
     011130  104300  003621  001530                                                  MOV     CURTRK,OUT.08
     011133  104300  002212  001523                                                  MOV     LUNIT,OUT.03
     011136  104202  105713                                                         MOV     #19!ERHARD+3000.,R2
     011140  104020  001522                                                         MOV     R2,OUT  2
     011142  104200  011142  001521                                                 MOV     #.,OUT.01
     011145  104200  060013  001520                                                 MOV     #ERRMES,OUT.RQ
  68 011150                                          ENDERR  19
     011150  104200  003276  001543                                                 MOV     #SER22,OUT.19
     011153  104200  000024  003626                                                 MOV     #19+1,ERRPOS    ; SET THE POSITION
  69 011156                                          CALL    TESTEX
     011156  020000  002372                          ^O20000,TESTEX
  70                                 ;               MOV     OUT.RQ,R0               ; PRINT ERROR
  71                                 ;               CALL    HOSTRQ
  72 011160                          FOREXT: RETURN
     011160  000000  000000                          ^O0,0
```

```
                                        .SBTTL  TRKTST - TEST TRACK
011162                          TRKTST:
                                ;
                                ;       TEST THE ENTIRE TRACK UNTIL AT LEAST ONE BLOCK IS SUCCESSFULLY
                                ;       WRITTEN AND READ WITH SEVERAL DATA PATTERNS
                                ;
011162  104307  003604                  MOV     TSTBLK,R0               ; MOVE LO STARTING DBN TO CURRENT DBN
011164  104070  003614                  MOV     R0,CURBLK
011166  104307  003605                  MOV     TSTBLK+1,R0             ; MOVE HI STARTING DBN TO CURRENT DBN
011170  104070  003615                  MOV     R0,CURBLK+1
011172  104301  003603                  MOV     SECTRK,R1               ; GET SECTORS/TRACK
011174  117401                          DEC     R1                      ; ADJUST FOR END LOOP WHEN NEGATIVE
011175  104010  003623          TRKLOP: MOV     R1,SECCNT               ; SAVE SECTORS LEFT IN SECTOR COUNT
011177  104201  000001                  MOV     #1,R1                   ; MOVE 1 TO CURRENT PATTERN
011201                                  CALL    WTRCMP                  ; WRITE THAN READ AND COMPARE THE SECTOR
011201  020000  011231                  ^O20000,WTRCMP
011203  115007                          TST     R0                      ; SEE IF WRITES, READS AND COMPARES WERE GOOD
011204                                  BEQ     TRKEXT                  ; IF SO, BRANCH
011204  010000  011227                  ^O10000,TRKEXT
011206  104307  003614                  MOV     CURBLK,R0               ; GET LO CURRENT BLOCK NUMBER
011210  115407                          INC     R0                      ; INCREMENT DBN NUMBER
011211  104070  003614                  MOV     R0,CURBLK               ; SAVE
011213                                  BCC     TRKBOT                  ; BRANCH IF NO CARRY
011213  040000  011222                  ^O40000,TRKBOT
011215  104307  003615                  MOV     CURBLK+1,R0             ; GET HI DBN NUMBER
011217  115407                          INC     R0                      ; INCREMENT
011220  104070  003615                  MOV     R0,CURBLK+1             ; SAVE
011222  104301  003623          TRKBOT: MOV     SECCNT,R1               ; GET SECTOR COUNT
011224  117401                          DEC     R1                      ; DECREMENT COUNT
011225                                  BPL     TRKLOP                  ; BRANCH IF COUNT POSITIVE
011225  030000  011175                  ^O30000,TRKLOP
011227                          TRKEXT: RETURN
011227  000000  000000                  ^O0,0
```

```
                                        .SBTTL  WRTCMP - WRITE A DATA PATTERN AND COMPARE
011231                          WTRCMP:
                                ;
                                ;       WRITE A DATA PATTERN TO A SECTOR, READ IT BACK, THEN DO A DATA
                                ;       COMPARE ON THE DATA.  DO THIS AS MANY TIMES AS THERE IS PATTERNS.
                                ;       CURRENT PATTERN NUMBER IN 'CURPAT'
                                ;
011231  104010  003622                MOV     R1,CURPAT               ; R1 IS CURRENT PATTERN NUMBER
011233  105201  003656                ADD     #PATPTR,R1              ; R1 NOW POINTS TO PATTERN TO GENERATE
011235                                CALL    GENPAT                  ; GENERATE THE PATTERN IN THE OUTPUT BUFFER
011235  020000  011267                ^O20000,GENPAT
011237                                CALL    WRITEB                  ; WRITE THE PATTERN
011237  020000  011314                ^O20000,WRITEB
011241  115007                        TST     R0                      ; SEE IF ANY ERROR OCCURRED
011242                                BNE     WTREXT                  ; IF SO, BRANCH
011242  050000  011265                ^O50000,WTREXT
011244                                CALL    READB                   ; READ THE BLOCK BACK
011244  020000  011602                ^O20000,READB
011246  115007                        TST     R0                      ; SEE IF ANY ERRORS OCCURRED
011247                                BNE     WTREXT                  ; IF SO, BRANCH
011247  050000  011265                ^O50000,WTREXT
011251                                CALL    CMPDAT                  ; COMPARE THE DATA
011251  020000  012041                ^O20000,CMPDAT
011253  115007                        TST     R0                      ; SEE IF ANY ERRORS OCCURRED
011254                                BNE     WTREXT                  ; IF SO, BRANCH
011254  050000  011265                ^O50000,WTREXT
011256  104301  003622                MOV     CURPAT,R1               ; GET CURRENT PATTERN
011260  115401                        INC     R1                      ; NEXT PATTERN
011261  106301  003656                CMP     PATPTR,R1               ; COMARE AGAINST MAXIMUM PATTERN NUMBER
011263                                BPL     WTRCMP                  ; IF LESS OR EQUAL, LOOP
011263  030000  011231                ^O30000,WTRCMP
011265                          WTREXT: RETURN
011265  000000  000000                ^O0,0
```

```
                                        .SBTTL  GENERATE A PATTERN
011267                          GENPAT:
                                ;
                                ;       GENERATE THE DATA PATTERN IN THE OUTPUT BUFFER
                                ;
                                ;       R1 POINTS TO POINTER TO PATTERN BUFFER WHICH IS PATTERN LENGTH
                                ;       (1 WORD) FOLLOWED BY THAT MANY WORDS OF PATTERN
                                ;
011267                                  PUSH    R2                      ; SAVE R2
011267  100462                                                                  MOV R2,-(SP)
011270  104117                          MOV     (R1),R0                 ; R0 POINTS TO START OF PATTERN BUFFER
011271  104203  000400                  MOV     #256.,R3                ; R3 HOLDS PATTERN COUNT
011273  104204  003767                  MOV     #OBUFF,R4               ; R4 POINTS TO OUTPUT BUFFER
011275  104071                  GENOUT: MOV     R0,R1                   ; R1 POINTS TO START OF PATTERN BUFFER
011276  104215                          MOV     (R1)+,R5                ; R5 CONTAINS LENGTH OF PATTERN
011277  104212                  GENIN:  MOV     (R1)+,R2                ; GET WORD OF PATTERN
011300  100242                          MOV     R2,(R4)+                ; MOVE TO BUFFER
011301  117403                          DEC     R3                      ; DECREMENT BUFFER COUNT
011302                                  BEQ     GENEXT                  ; IF BUFFER FULL, EXIT
011302  010000  011311                  ^O10000,GENEXT
011304  117405                          DEC     R5                      ; DECREMENT PATTERN COUNT
011305                                  BNE     GENIN                   ; IF NON-ZERO, BRANCH
011305  050000  011277                  ^O50000,GENIN
011307                                  BR      GENOUT                  ; START PATTERN OVER AGAIN
011307  000000  011275                  ^O0,GENOUT
011311                          GENEXT: POP     R2                      ; RESTORE R2
011311  104262                                                                  MOV (SP)+,R2
011312                                  RETURN
011312  000000  000000                  ^O0,0
```

```
                                                .SBTTL  WRITEB - WRITE A SECTOR
011314                                  WRITEB:
                                        ;
                                        ;       WILL WRITE ONE SECTOR OF DATA TO THE DISK
                                        ;
011314  104207  140000                          MOV     #WSTOP,R0
011316  104070  003647                          MOV     R0,RW.CPT+CHAIN
011320  104070  003650                          MOV     R0,RW.STAT+CHAIN; MOVE END-OF-CHAIN TO CHAIN
011322  104307  003614                          MOV     CURBLK,R0               ; MOVE LO DBN TO CHAIN
011324  104070  003652                          MOV     R0,RW.LOW+CHAIN
011326  104307  003615                          MOV     CURBLK+1,R0             ; MOVE HI DBN TO CHAIN
011330  104301  003521                          MOV     SUB+HIDBN,R1            ; GET HIGH ORDER BITS OF STARTING DBN
011332  110601                                  ROR     R1                      ; MOVE TO CORRECT POSITION
011333  110601                                  ROR     R1
011334  110601                                  ROR     R1
011335  110601                                  ROR     R1
011336  103201  170377                          BIC     #HBLONB,R1              ; CLEAR UNUSED BITS
011340  101201  140000                          BIS     #HD.DBN,R1              ; SET HEADER CODE
011342  101017                                  BIS     R1,R0                   ; SET IN R0
011343  104070  003653                          MOV     R0,RW.HI+CHAIN
011345  104307  003621                          MOV     CURTRK,R0               ; MOVE TRACK + RTC TO CHAIN
011347  101207  122400                          BIS     #WREAL,R0               ; SET REAL TIME COMMAND WRITTE
011351  104070  003654                          MOV     R0,RW.CMD+CHAIN
011353  104207  003767                          MOV     #OBUFF,R0               ; MOVE OUTPUT BUFFER TO CHAIN
011355  104070  003651                          MOV     R0,RW.BUF+CHAIN
011357  104302  003356                          MOV     SDI,R2                  ; MOVE MASK TO R2
011361                                          CALL    RDSTAT                  ; GET DRIVE STATUS
011361  020000  001432                          ^020000,RDSTAT
011363  115002                                  TSTB    R2                      ; WAS IT OK?
011364                                          BEQ     2$                      ; IF NO ERROR, BRANCH
011364  010000  011436                          ^010000,2$
011366  102201  000400                          BIT     #RCVERR,R1              ; SEE IF ANY ERRORS
011370                                          BNE     1$                      ; IF NOT, ERROR
011370  050000  011414                          ^050000,1$
011372                                          HARDER  10                      ; REPORT INVALID STATUS ERROR
011372  104200  000341  001524                          MOV     #MS10,OUT.04
011375  104300  002212  001523                          MOV     LUNIT,OUT.03
011400  104202  105702                                  MOV     #10!ERHARD+3000.,R2
011402  104020  001522                                  MOV     R2,OUT.02
011404  104200  011404  001521                          MOV     #.,OUT.01
011407  104200  060013  001520                          MOV     #ERRMES,OUT.RQ
011412                                          BR      TESTEW                  ; BRANCH TO DONE
011412  000000  002355                          ^00,TESTEW
011414                                  1$:
011414                                          HARDER  12                      ; REPORT XMIT ERROR
011414  104200  000434  001524                          MOV     #MS12,OUT.04
011417  104300  002212  001523                          MOV     LUNIT,OUT.03
011422  104202  105704                                  MOV     #12!ERHARD+3000.,R2
011424  104020  001522                                  MOV     R2,OUT.02
011426  104200  011426  001521                          MOV     #.,OUT.01
011431  104200  060013  001520                          MOV     #ERRMES,OUT.RQ
011434                                          BR      TESTEW                  ; BRANCH TO DONE
011434  000000  002355                          ^00,TESTEW
011436                                  2$:
011436  102201  100000                          BIT     #RWRDY,R1               ; SEE IF READ/WRITE READY IS ASSERTED
011440                                          BNE     WGO                     ; IF SO, BRANCH
011440  050000  011472                          ^050000,WGO
```

```
     46 011442  104307  003623                    MOV     SECCNT,R0           ; GET SECTOR COUNT
     47 011444                                    BNE     WRTEXT              ; IF NONZERO, DO NOT REPORT ERROR
        011444  050000  011600                    ^050000,WRTEXT
     48 011446                                    HARDER  22                  ;READ/WRITE DROPPED READY BEFORE WRITE
        011446  104200  001117  001524                                                MOV     #MS22,OUT.04
        011451  104300  002212  001523                                                MOV     LUNIT,OUT.03
        011454  104202  105716                                                        MOV     #22!ERHARD+3000.,R2
        011456  104020  001522                                                        MOV     R2,OUT.02
        011460  104200  011460  001521                                                MOV     #.,OUT.01
        011463  104200  060013  001520                                                MOV     #ERRMES,OUT.RQ
     49 011466  104207  000001                    MOV     #1,R0               ; FLAG ERROR
     50 011470                                    BR      WRTEXT              ; BRANCH
        011470  000000  011600                    ^00,WRTEXT
     51 011472  104302  003356          WGO:      MOV     SDI,R2              ; SET PORT INDICATOR
     52 011474  060012                            XFC     WAITSI              ; WAIT FOR SECTOR OR INDEX PULSE
     53 011475  104207  003647                    MOV     #CHAIN,R0           ; POINT TO WRITE CHAIN
     54 011477  104304  003523                    MOV     SUB+DATPRE,R4       ; MOVE DATA PREAMBLE LENGTH TO R4
     55 011501  103204  177400                    BIC     #HIBYTE,R4          ; CLEAR UNUSED BITS
     59 011503  104202  000400                    MOV     #BUFFSZ,R2          ; R2 = BUFFER SIZE FOR 52
     63 011505  060003                            XFC     XWRITE              ; WRITE THE BLOCK
     64 011506  114007                            CLR     R0                  ; FLAG AS NO ERRORS
     65 011507  115000  003627                    TST     WFLAG               ; DID WE TEST WHEN WRITE PROTECTED?
     66 011511                                    BNE     WRTEXT              ; IF SO, EXIT
        011511  050000  011600                    ^050000,WRTEXT
     67 011513  115001                            TST     R1                  ; SEE IF AN ERROR OCCURRED
     68 011514                                    BEQ     WRTEXT              ; IF NOT, BRANCH
        011514  010000  011600                    ^010000,WRTEXT
     69 011516  104307  003623                    MOV     SECCNT,R0           ; SEE IF ERROR SHOULD BE REPORTED
     70 011520                                    BNE     WRTEXT              ; IF NOT, BRANCH
        011520  050000  011600                    ^050000,WRTEXT
     71 011522                                    HARDER  23,<R1,CURBLK,CURBLK+1,INS+1,INS+2,INS+3,CURTRK> ;ERROR DURING WRITE
        011522  104200  001151  001524                                                MOV     #MS23,OUT.04
        011525  104010  001525                                                        MOV     R1,OUT.05
        011527  104300  003614  001526                                                MOV     CURBLK,OUT.06
        011532  104300  003615  001527                                                MOV     CURBLK+1,OUT.07
        011535  104300  003461  001530                                                MOV     INS+1,OUT.08
        011540  104300  003462  001531                                                MOV     INS+2,OUT.09
        011543  104300  003463  001532                                                MOV     INS+3,OUT.10
        011546  104300  003621  001533                                                MOV     CURTRK,OUT.11
        011551  104300  002212  001523                                                MOV     LUNIT,OUT.03
        011554  104202  105717                                                        MOV     #23!ERHARD+3000.,R2
        011556  104020  001522                                                        MOV     R2,OUT.02
        011560  104200  011560  001521                                                MOV     #.,OUT.01
        011563  104200  060013  001520                                                MOV     #ERRMES,OUT.RQ
     72 011566                                    ENDERR  12
        011566  104200  003276  001534                                                MOV     #SER22,OUT.12
        011571  104200  000015  003626                                                MOV     #12+1,ERRPOS    ; SET THE POSITION
     73 011574                                    CALL    TESTEX
        011574  020000  002372                    ^020000,TESTEX
     74                                 ;         MOV     OUT.RQ,R0
     75                                 ;         CALL    HOSTRQ
     76 011576  104207  000001                    MOV     #1,R0               ; MAKE R0 NONZERO TO REPORT ERROR
     77 011600                          WRTEXT:   RETURN
        011600  000000  000000                    ^00,0
```

```
                                                .SBTTL  READB - READ ONE SECTOR
011602                                  READB:
                                        ;
                                        ;       READ ONE SECTOR FROM DISK
                                        ;
011602  104207  100000                          MOV     #RSTOP,R0               ; MOVE END-OF-CHAIN TO CHAIN
011604  104070  003647                          MOV     R0,RW.CPT+CHAIN
011606  104070  003650                          MOV     R0,RW.STAT+CHAIN
011610  104307  003621                          MOV     CURTRK,R0               ; MOVE TRACK NUMBER AND RTC TO CHAIN
011612  101207  013400                          BIS     #RREAL,R0               ; SET REAL TIME READ COMMMAND
011614  104070  003654                          MOV     R0,RW.CMD+CHAIN
011616  104207  004377                          MOV     #RBUFD,R0               ; MOVE POINTER TO INPUT BUFFER INTO CHAIN
011620  104070  003651                          MOV     R0,RW.BUF+CHAIN
011622  104302  003356                          MOV     SDI,R2                  ; MOVE MASK TO R2
011624                                          CALL    RDSTAT                  ; GET DRIVE STATUS
011624  020000  001432                          ^O20000,RDSTAT
011626  115002                                  TST     R2                      ; WAS IT OK?
011627                                          BEQ     2$                      ; IF NO ERROR, BRANCH
011627  010000  011701                          ^O10000,2$
011631  102201  000400                          BIT     #RCVERR,R1              ; SEE IF ANY ERRORS
011633                                          BNE     1$                      ; IF NOT, ERROR
011633  050000  011657                          ^O50000,1$
011635                                          HARDER  10                      ; REPORT INVALID STATUS ERROR
011635  104200  000341  001524                                                          MOV     #MS10,OUT.04
011640  104300  002212  001523                                                          MOV     LUNIT,OUT.03
011643  104202  105702                                                                  MOV     #10!ERHARD+3000.,R2
011645  104020  001522                                                                  MOV     R2,OUT.02
011647  104200  011647  001521                                                          MOV     #.,OUT.01
011652  104200  060013  001520                                                          MOV     #ERRMES,OUT.RQ
011655                                          BR      TESTEW                  ; BRANCH TO DONE
011655  000000  002355                          ^O0,TESTEW
011657                                  1$:
011657                                          HARDER  12                      ; REPORT XMIT ERROR
011657  104200  000434  001524                                                          MOV     #MS12,OUT.04
011662  104300  002212  001523                                                          MOV     LUNIT,OUT.03
011665  104202  105704                                                                  MOV     #12!ERHARD+3000.,R2
011667  104020  001522                                                                  MOV     R2,OUT.02
011671  104200  011671  001521                                                          MOV     #.,OUT.01
011674  104200  060013  001520                                                          MOV     #ERRMES,OUT.RQ
011677                                          BR      TESTEW                  ; BRANCH TO DONE
011677  000000  002355                          ^O0,TESTEW
011701                                  2$:
011701  102201  100000                          BIT     #RWRDY,R1               ; SEE IF READ/WRITE READY IS ASSERTED
011703                                          BNE     RGO                     ; IF SO, BRANCH
011703  050000  011741                          ^O50000,RGO
011705  104307  003623                          MOV     SECCNT,R0               ; GET SECTOR COUNT
011707                                          BNE     REDEXT                  ; IF NONZERO, DO NOT REPORT ERROR
011707  050000  012037                          ^O50000,REDEXT
011711                                          HARDER  24                      ;READ/WRITE DROPPED READY BEFORE READ
011711  104200  001306  001524                                                          MOV     #MS24,OUT.04
011714  104300  002212  001523                                                          MOV     LUNIT,OUT.03
011717  104202  105720                                                                  MOV     #24!ERHARD+3000.,R2
011721  104020  001522                                                                  MOV     R2,OUT.02
011723  104200  011723  001521                                                          MOV     #.,OUT.01
011726  104200  060013  001520                                                          MOV     #ERRMES,OUT.RQ
011731                                          ENDERR  5
011731  104200  003276  001525                                                          MOV     #SER22,OUT.05
```

```
      011734  104200  000006  003626                                          MOV     #5+1,ERRPOS     ; SET THE POSITION
   38 011737                                  BR      REDERR                  ; GO PRINT THE ERROR
      011737  000000  012033                  ^O0,REDERR
   39 011741  104302  003356          RGO:    MOV     SDI,R2                  ; SET PORT INDICATOR
   40 011743  060012                          XFC     WAITSI                  ; WAIT FOR SECTOR OR INDEX PULSE
   41 011744  104207  003647                  MOV     #CHAIN,R0               ; POINT TO READ CHAIN
   45 011746  104202  000400                  MOV     #BUFFSZ,R2              ; R2 = BUFFER SIZE FOR 52
   49 011750  060002                          XFC     XREAD                   ; READ THE SECTOR
   50 011751  114007                          CLR     R0                      ; FLAG AS NO ERRORS
   51 011752  115001                          TST     R1                      ; SEE IF ERROR OCCURRED
   52 011753                                  BEQ     REDEXT                  ; IF NOT, BRANCH
      011753  010000  012037                  ^O10000,REDEXT
   53 011755  104307  003623                  MOV     SECCNT,R0               ; SEE IF ERROR SHOULD BE REPORTED
   54 011757                                  BNE     REDEXT                  ; IF NOT, BRANCH
      011757  050000  012037                  ^O50000,REDEXT
   55 011761                                  HARDER  25,<R1,CURBLK,CURBLK+1,INS+1,INS+2,INS+3,CURTRK> ;ERROR DURING READ
      011761  104200  001337  001524                                          MOV     #MS25,OUT.04
      011764  104010  001525                                                  MOV     R1,OUT.05
      011766  104300  003614  001526                                          MOV     CURBLK,OUT.06
      011771  104300  003615  001527                                          MOV     CURBLK+1,OUT.07
      011774  104300  003461  001530                                          MOV     INS+1,OUT.08
      011777  104300  003462  001531                                          MOV     INS+2,OUT.09
      012002  104300  003463  001532                                          MOV     INS+3,OUT.10
      012005  104300  003621  001533                                          MOV     CURTRK,OUT.11
      012010  104300  002212  001523                                          MOV     LUNIT,OUT.03
      012013  104202  105721                                                  MOV     #25!ERHARD+3000.,R2
      012015  104020  001522                                                  MOV     R2,OUT.02
      012017  104200  012017  001521                                          MOV     #.,OUT.01
      012022  104200  060013  001520                                          MOV     #ERRMES,OUT.RQ
   56 012025                                  ENDERR  12
      012025  104200  003276  001534                                          MOV     #SER22,OUT.12
      012030  104200  000015  003626                                          MOV     #12+1,ERRPOS    ; SET THE POSITION
   57 012033                          REDERR: CALL    TESTEX
      012033  020000  002372                  ^O20000,TESTEX
   58                                 ;       MOV     OUT.RQ,R0
   59                                 ;       CALL    HOSTRQ
   60 012035  104207  000001                  MOV     #1,R0                   ; FLAG ERROR
   61 012037                          REDEXT: RETURN
      012037  000000  000000                  ^O0,0
```

```
      1                                                .SBTTL  CMPDAT - COMPARE DATA
      2 012041                                 CMPDAT:
      3                                                ;
      4                                                ;       COMPARE THE DATA IN 'OBUFF' (OUTPUT BUFFER) WITH 'RBUFD'
      5                                                ;       (INPUT BUFFER)
      6                                                ;
      7 012041  104207  003767                       MOV     #OBUFF,R0               ; R0 POINTS AT OUTPUT BUFFER
      8 012043  104201  004377                       MOV     #RBUFD,R1               ; R1 POINTS AT INPUT BUFFER
      9 012045  114003                               CLR     R3                      ; COUNTER
     10 012046  104274                       CMPLOP: MOV     (R0)+,R4                ; GET OUTPUT BUFFER WORD
     11 012047  106214                               CMP     (R1)+,R4                ; COMPARE AGAINST INPUT BUFFER
     12 012050                                       BEQ     NOERR                   ; IF NO ERROR, BRANCH
        012050  010000  012130                       ^O010000,NOERR
     13 012052  104305  003623                       MOV     SECCNT,R5               ; SEE IF ERROR IS TO BE REPORTED
     14 012054                                       BNE     CMPEXT                  ; IF NOT, BRANCH
        012054  050000  012136                       ^O050000,CMPEXT
     15 012056                                       HARDER  26,<R3,-(R0),-(R1),INS+1,INS+2,INS+3,CURTRK> ;DATA COMPARE FAILURE
        012056  104200  001473  001524                                                       MOV     #MS26,OUT.04
        012061  104030  001525                                                              MOV     R3,OUT.05
        012063  104470  001526                                                              MOV     -(R0),OUT.06
        012065  104410  001527                                                              MOV     -(R1),OUT.07
        012067  104300  003461  001530                                                      MOV     INS+1,OUT.08
        012072  104300  003462  001531                                                      MOV     INS+2,OUT.09
        012075  104300  003463  001532                                                      MOV     INS+3,OUT.10
        012100  104300  003621  001533                                                      MOV     CURTRK,OUT.11
        012103  104300  002212  001523                                                      MOV     LUNIT,OUT.03
        012106  104202  105722                                                             MOV     #26!ERHARD+3000.,R2
        012110  104020  001522                                                             MOV     R2,OUT.02
        012112  104200  012112  001521                                                     MOV     #.,OUT.01
        012115  104200  060013  001520                                                     MOV     #ERRMES,OUT.RQ
     16 012120  104307  001520                       MOV     OUT.RQ,R0
     17 012122                                       CALL    HOSTRQ
        012122  020000  001463                       ^O020000,HOSTRQ
     18 012124  104207  000001                       MOV     #1,R0                   ; SET FOR ERROR
     19 012126                                       BR      CMPEXT                  ; EXIT
        012126  000000  012136                       ^O00,CMPEXT
     20 012130  115403                       NOERR:  INC     R3                      ; INCREMENT COUNT
     21 012131  106203  000400                       CMP     #256.,R3                ; SEE IF COMPARE COMPLETE
     22 012133                                       BNE     CMPLOP                  ; IF NOT, BRANCH
        012133  050000  012046                       ^O050000,CMPLOP
     23 012135  114007                               CLR     R0                      ; FLAG AS NO ERRORS
     24 012136                               CMPEXT: RETURN
        012136  000000  000000                       ^O00,0
     25
     43 012140                                       MESSAGES
        012140  021507                               .WREDC                  ;OUTPUT EDC FOR THIS OVERLAY
        000000     042     124     111  MS1:        .ASCII\"TIME-OUT ON SEND'N\
        000011     122     061     122              .ASCII\R1R1\
        000013     000                              .BYTE 0
        000014     042     124     111  MS2:        .ASCII\"TIME-OUT ON RECEIVE'N\
        000027     122     061     122              .ASCII\R1R1\
        000031     000                              .BYTE 0
        000032     042     106     111  MS3:        .ASCII\"FIRST WORD RECEIVED WAS NOT A START FRAME'N\
        000060     122     061     122              .ASCII\R1R1\
        000062     000                              .BYTE 0
        000063     042     106     122  MS4:        .ASCII\"FRAMING ERROR ON LEVEL 0 RESPONSE'N\
        000105     122     061     122              .ASCII\R1R1\
```

```
000107  000                             .BYTE 0
000110  042  103  110  MS5:    .ASCII\"CHECKSUM ERROR ON LEVEL 0 RESPONSE'N\
000132  122  061  122                  .ASCII\R1R1\
000134  000                             .BYTE 0
000135  042  122  105  MS6:    .ASCII\"RESPONSE LONGER THAN EXPECTED'N\
000155  122  061  122                  .ASCII\R1R1\
000157  000                             .BYTE 0
000160  042  103  117  MS7:    .ASCII\"CODE FROM RECEIVE WAS UNINTELLIGIBLE FROM SUBSYSTEM = 'H16N\
000216  122  061  122                  .ASCII\R1R1\
000220  000                             .BYTE 0
000221  042  103  117  MS8:    .ASCII\"COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
000251  122  061                       .ASCII\R1\
000252  042  040  040             .ASCII\"  EXPECTED RESPONSE  'H8N\
000267  042  040  040             .ASCII\"  ACTUAL RESPONSE    'H8N\
000304  122  061                       .ASCII\R1\
000305  000                             .BYTE 0
000306  042  104  122  MS9:    .ASCII\"DRIVE NOT ASSERTING RECEIVER READY IN DRIVE STATE'N\
000340  000                             .BYTE 0
000341  042  106  101  MS10:   .ASCII\"FAILED TO RECEIVE VALID DRIVE STATE'NR1\
000365  000                             .BYTE 0
000366  042  103  101  MS11:   .ASCII\"CANNOT RECEIVE DRIVE STATE FROM DRIVE'N\
000412  042  103  110             .ASCII\"CHECK IF DRIVE IS POWERED ON.'NR1\
000433  000                             .BYTE 0
000434  042  104  122  MS12:   .ASCII\"DRIVE STATE RECEIVED HAS BAD PARITY'NR1\
000460  000                             .BYTE 0
000461  042  116  117  MS13:   .ASCII\"NO VALID STATE FROM DRIVE'NR1\
000500  000                             .BYTE 0
000501  042  123  125  MS14:   .ASCII  \"SUBUNIT CHARACTERISTICS SAY THERE ARE ZERO READ ONLY GROUPS'N\
000540  042  111  116             .ASCII  \"IN THE DIAGNOSTIC AREA'N\
000554  000                             .BYTE   0
000555  042  123  125  MS15:   .ASCII  \"SUBUNIT CHARACTERISTICS SAY THERE ARE LESS THAN 1 READ/WRITE'N\
000614  042  107  122             .ASCII  \"GROUPS IN THE DIAGNOSTIC AREA'N\
000634  000                             .BYTE   0
000635  042  116  105  MS16:   .ASCII\"NEITHER R/W READY NOR ATTENTION SET AFTER RECALIBRATE COMMAND'NR1\
000676  000                             .BYTE 0
000677  042  123  125  MS17:   .ASCII\"SUBUNIT CHARACTERISTICS SAY LESS THAN 1 DIAGNOSTIC CYLINDER'N\
000736  000                             .BYTE 0
000737  042  122  105  MS18:   .ASCII\"READ/WRITE READY DROPPED BEFORE FORMAT OPERATION'N\
000770  000                             .BYTE 0
000771  042  106  117  MS19:   .ASCII\"FORMAT OPERATION REPORTED TIME-OUT FAILURE'N\
001017  042  040  040             .ASCII\"  CYLINDER 'D28".   GROUP  'D8".  TRACK 'D8"."N\
001047  000                             .BYTE 0
001050  042  101  106  MS20:   .ASCII\"AFTER RECAL, ERROR BITS WERE SET'NR1\
001072  000                             .BYTE 0
001073  116  042  114  MS21:   .ASCII\N"LOGGABLE INFORMATION AFTER RECAL'NR1\
001116  000                             .BYTE 0
001117  042  122  105  MS22:   .ASCII\"READ/WRITE READY DROPPED BEFORE WRITE OPERATION'N\
001150  000                             .BYTE 0
001151  042  103  117  MS23:   .ASCII\"COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
001212  042  127  122             .ASCII\"WRITE OPERATION REPORTED FAILURE -- ERROR CODE 'D8" OCTAL."N\
001251  042  104  102             .ASCII\"DBN 'D24".  CYLINDER 'D28".  GROUP 'D8".  TRACK 'D8"."N\
001305  000                             .BYTE 0
001306  042  122  105  MS24:   .ASCII\"READ/WRITE READY DROPPED BEFORE READ OPERATION'N\
001336  000                             .BYTE 0
001337  042  103  117  MS25:   .ASCII\"COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
001400  042  122  105             .ASCII\"READ OPERATION REPORTED FAILURE -- ERROR CODE 'D8" OCTAL."N\
001436  042  104  102             .ASCII\"DBN 'D24".  CYLINDER 'D28".  GROUP 'D8".  TRACK 'D8"."N\
```

```
001472  000                             .BYTE 0
001473  042  103  117  MS26:   .ASCII\'COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
001534  042  104  101          .ASCII\'DATA COMPARE FAILURE ON WORD 'D16'.'N\
001557  042  105  130          .ASCII\'EXPECTED DATA  'H16N\
001572  042  101  103          .ASCII\'ACTUAL DATA    'H16N\
001604  042  103  131          .ASCII\'CYLINDER 'D28'.  GROUP 'D8'.  TRACK 'D8'.'N\
001632  000                     .BYTE 0
001633  042  123  105  MS27:   .ASCII\'SEEK COMPLETE TIME-OUT -- READ/WRITE READY DID NOT SET.'N\
001670  042  123  105          .ASCII\'SEEK WAS TO CYLINDER 'D28'.  GROUP 'D8'.'N\
001715  000                     .BYTE 0
001716  042  116  117  MS28:   .ASCII\'NO BLOCK ON THIS TRACK CAN BE READ. LAST BLOCK TRIED:'N\
001752  101  061  042          .ASCII\A1'BN 'D24'.  CYLINDER 'D28'.  GROUP 'D8'.  TRACK 'D8'.'N\
002006  000                     .BYTE 0
002007  042  101  126  MS29:   .ASCII  \'AVAILABLE WAS NOT ASSERTED AFTER DISCONNECT'N\
002036  042  040  040          .ASCII  \'  STATE RECEIVED  'H16N\
002052  000                     .BYTE   0
002053  042  111  116  MS30:   .ASCII\'INVALID COMMAND 'H16' WAS SUCCESSFUL'N\
002076  000                     .BYTE   0
002077  042  103  117  MS31:   .ASCII\'COMMAND WITH 'R1' LENGTH = 'D8' WAS SUCCESSFUL'N\
002127  000                     .BYTE   0
002130  042  125  116  MS32:   .ASCII\'UNIT DID NOT REPORT TRANSMITTION ERROR'NR1\
002155  000                     .BYTE   0
002156  042  125  116  MS33:   .ASCII\'UNIT ACCEPTED AN INVALID GROUP NUMBER FROM GROUP SELECT LEVEL 1'N\
002217  000                     .BYTE   0
002220  042  125  116  MS34:   .ASCII\'UNABLE TO CORRECTLY READ OVERLAY 'O3NR1\
002244  000                     .BYTE   0
002245  042  123  125  MS35:   .ASCII\'SUCCESSFULLY WROTE IN DBN AREA WHEN DRIVE WAS WRITE PROTECTED'N\
002305  000                     .BYTE   0
002306  042  104  122  MS36:   .ASCII  \'DRIVE IS NOT PROPERLY FORMATTED.'NR1\
002330  000                     .BYTE   0
002331  042  104  122  MS37:   .ASCII  \'DRIVE IS FORMATTED IN 576 BYTE MODE.'N\
002354  042  124  117          .ASCII  \'TO RUN WITH A UDA, THIS DRIVE NEEDS TO BE FORMATTED '\
002407  042  111  116          .ASCII  \'IN 512 BYTE MODE.'N\
002421  000                     .BYTE   0
002422  042  116  117  MS38:   .ASCII  \'NO COPY OF THE FCT COULD BE READ.'NR1\
002445  000                     .BYTE   0
002446  042  125  104  SEP36:  .ASCII  \'UDA WILL SPIN DOWN THIS DRIVE IF USED IN NORMAL SYSTEM OPERATION.'N\
002510  042  124  110          .ASCII  \'THIS DRIVE NEEDS TO BE FORMATTED.'\
002531  000                     .BYTE   0
002532  042  124  110  SER39:  .ASCII\'THIS UDA AND ALL DRIVES ATTACHED WILL BE REMOVED FROM TESTING'N\
002572  000                     .BYTE   0
002573  042  103  117  SER00:  .ASCII\'COMMAND WAS 'R1\
002603  000                     .BYTE   0
002604  042  117  116  MS.ONL: .ASCII\'ONLINE'N\
002610  000                     .BYTE   0
002611  042  104  122  MS.CLR: .ASCII\'DRIVE CLEAR'N\
002620  000                     .BYTE   0
002621  042  104  111  MS.DIS: .ASCII\'DISCONNECT'N\
002627  000                     .BYTE   0
002630  042  107  105  MS.GCR: .ASCII\'GET COMMON CHARACTERISTICS'N\
002646  000                     .BYTE   0
002647  042  107  105  MS.SCR: .ASCII\'GET SUBUNIT CHARACTERISTICS'N\
002666  000                     .BYTE   0
002667  042  107  105  MS.GST: .ASCII\'GET STATUS'N\
002675  000                     .BYTE   0
002676  042  103  110  MS.MOD: .ASCII\'CHANGE MODE'N\
002705  000                     .BYTE   0
```

```
002706  042  123  105  MS.SEK:  .ASCII\''SEEK''N\
002711  000                    .BYTE   0
002712  042  111  116  MS.INR:  .ASCII\''INITIATE RECALIBRATE''N\
002725  000                    .BYTE   0
002726  042  123  120  MS.RUN:  .ASCII\''SPIN UP''N\
002733  000                    .BYTE   0
002734  042  125  116  MS2000:  .ASCII  \''UNABLE FIND REQUESTED DRIVE FOR TESTING''N\
002761  042  124  110           .ASCII  \''THE FOLLOWING IS VISIBLE ON THE PORTS''N\
003005  042  125  104           .ASCII  \''UDA PORT 0 -- ''R1\
003016  042  125  104           .ASCII  \''UDA PORT 1 -- ''R1\
003027  042  125  104           .ASCII  \''UDA PORT 2 -- ''R1\
003040  042  125  104           .ASCII  \''UDA PORT 3 -- ''R1\
003051  000                    .BYTE   0
003052  042  116  117  SER10:   .ASCII  \''NO DRIVE ATTACHED''N\
003064  000                    .BYTE   0
003065  042  122  103  SER11:   .ASCII  \''RCVR RDY NEVER ASSERTED''N\
003102  000                    .BYTE   0
003103  042  124  111  SER12:   .ASCII  \''TIMEOUT OF SEND''N\
003114  000                    .BYTE   0
003115  042  124  111  SER13:   .ASCII  \''TIMEOUT OF RECEIVE''N\
003127  000                    .BYTE   0
003130  042  106  111  SER14:   .ASCII  \''FIRST WORD RECEIVED WAS NOT START FRAME''N\
003155  000                    .BYTE   0
003156  042  106  122  SER15:   .ASCII  \''FRAMING ERROR ON LEVEL 0 RECEIVE''N\
003177  000                    .BYTE   0
003200  042  103  110  SER16:   .ASCII  \''CHECKSUM ERROR ON LEVEL 0 RECEIVE''N\
003222  000                    .BYTE   0
003223  042  122  105  SER17:   .ASCII  \''RESPONSE LONGER THAN EXPECTED FOR GET STATUS CMD''N\
003254  000                    .BYTE   0
003255  042  104  122  SER18:   .ASCII  \''DRIVE ''R1\
003262  000                    .BYTE   0
003263  104  066  042  SER18D:  .ASCII  \D6'', ''\
003266  104  066  042  SER18C:  .ASCII  \D6'', ''\
003271  104  066  042  SER18B:  .ASCII  \D6'', ''\
003274  104  066  116  SER18A:  .ASCII  \D6N\
003275  000                    .BYTE   0
003276  042  122  105  SEP22:   .ASCII\''REAL TIME STATE ''H16N\
003311  042  123  124           .ASCII\''STATUS (R TO L): ''H16S2H16S2H16S2H16S2H16S2H16S2H16N\
003343  000                    .BYTE   0
003344  042  122  105  SER23:   .ASCII\''REAL TIME STATE ''H16N\
003357  000                    .BYTE   0
003360  042  104  122  SER40:   .ASCII  \''DRIVE NOT AVAILABLE TO THIS UDA''N\
003401  000                    .BYTE   0
003402  042  104  122  SER41:   .ASCII  \''DRIVE NOT SPINABLE''N\
003414  000                    .BYTE   0
003415  042  103  117  SER50:   .ASCII\''COMMAND''\
003421  000                    .BYTE   0
003422  042  122  105  SER51:   .ASCII\''RESPONSE''\
003427  000                    .BYTE   0
003430  042  127  110  SER52:   .ASCII\''WHEN A CONTINUE OR END FRAME DID NOT FOLLOW A START FRAME''N\
003466  000                    .BYTE   0
003467  042  127  110  SER53:   .ASCII\''WHEN AN END FRAME WAS SENT WITH NO START FRAME''N\
003517  000                    .BYTE   0
003520  042  127  110  SER54:   .ASCII\''WHEN AN END FRAME WITH A BAD CHECKSUM WAS SENT''N\
003550  000                    .BYTE   0
003551  042  127  110  SER55:   .ASCII\''WHEN A CONTINUE FRAME WAS SENT WITH NO START FRAME''N\
003603  000                    .BYTE   0
```

MESSAGES

```
        003604      042     127     110  SER56:  .ASCII\'WHEN TWO CONSECUTIVE START FRAMES WERE SENT'N\
        003633      000                          .BYTE   0
        003634      042     127     110  SER57:  .ASCII\'WHEN AN END FRAME WAS SENT AFTER A START FRAME TIMED OUT'N\
        003671      000                          .BYTE   0
     47 003672                                   DMEND
        003672  112505                           .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
     48         000001                           .END
```

SYMBOL TABLE

```
AREA    003624  DRVRUN= 000014  GRPOFF= 000011  MS.DIS  002621  OUT.RQ  001520
ATTN  = 000002  DR.CLR  002407  GST     003455  MS.END= 131000  OUT.01  001521
AVAIL = 000100  ECC    = 000015  HBHINB= 007777  MS.GCR 002630  OUT.02  001522
BF.DAT= 000000  ECCFLG= 010000  HBLONB= 170377  MS.GST 002667  OUT.03  001523
BF.ECC= 000401  ECCRSH= 000002  HDRPRE= 000005  MS.INR 002712  OUT.04  001524
BF.EDC= 000400  ECHO   = 000010  HD.BAD= 110000  MS.MOD 002676  OUT.05  001525
BLOCKC  003613  ECHOC  = 000350  HD.DBN= 140000  MS.ONL 002604  OUT.06  001526
BLOCKG  003612  ENDGTU  005040  HD.LBN= 000000  MS.RUN 002726  OUT.07  001527
BLOCKT  003611  ENDPNT= 005774  HD.PRV= 050000  MS.SCR 002647  OUT.08  001530
BREAK = 000000  EOC    = 100000  HD.RBN= 060000  MS.SEK 002706  OUT.09  001531
BUFFLG= 040000  ERECOV= 000006  HD.REV= 030000  MS.STR= 070400  OUT.10  001532
BUFFSZ= 000400  ERHARD= 100000  HD.XBN= 120000  MS1     000000  OUT.11  001533
BUFSIZ= 000043  ERLEV  = 000002  HEADER= 000002  MS10    000341  OUT.12  001534
CHAIN   003647  ERRMC  = 060014  HIBYTE= 177400  MS11    000366  OUT.13  001535
CHECK = 000010  ERRMES= 060013  HICYL  = 000001  MS12    000434  OUT.14  001536
CHGMOD= 000201  ERRN   = 005670  HIDBN  = 000003  MS13    000461  OUT.15  001537
CHRRES= 000170  ERRORS  003451  HILBN  = 000002  MS14    000501  OUT.16  001540
CMPDAT  012041  ERRPOS  003626  HIMEM  = 037777  MS15    000555  OUT.17  001541
CMPEXT  012136  ERSOFT= 140000  HIRBN  = 000003  MS16    000635  OUT.18  001542
CMPLOP  012046  EXIT   = 000021  HIXBN  = 000002  MS17    000677  OUT.19  001543
COMND   003357  FB.DAT= 000000  HOSTRQ  001463  MS18    000737  OUT.20  001544
COMPAR= 000006  FB.EDC= 000400  HRDREV= 000003  MS19    000771  OUT.21  001545
COMPLT= 000176  FCARY   010756  INDEXS  003646  MS2     000014  OUT.22  001546
COPY    003471  FCHAIN  004377  INR     003457  MS20    001050  OUT.23  001547
CR      003503  FCTSIZ= 000010  INS     003460  MS2000  002734  OUT.24  001550
CR.CLR  003372  FDIACY  003577  INSEEK= 000012  MS21    001073  OUT.25  001551
CR.DIS  003377  FGO     011071  INTEDC= 000105  MS22    001117  OUT.26  001552
CR.GCR  003365  FLAG    003630  IRECLB= 000216  MS23    001151  OUT.27  001553
CR.GST  003411  FLOOP   010747  LARGE  = 000001  MS24    001306  OUT.28  001554
CR.INR  003430  FNDCYL  002420  LBHINB= 177417  MS25    001337  OUT.29  001555
CR.MOD  003416  FNDCY2  002427  LBLONB= 177760  MS26    001473  OUT.30  001556
CR.ONL  003360  FND3    002440  LBNCYL= 000000  MS27    001633  OUT.31  001557
CR.RUN  003435  FND4    002470  LBNHST= 000012  MS28    001716  OUT.32  001560
CR.SCR  003404  FOREXT  011160  LBNTRK= 000011  MS29    002007  OUT.33  001561
CR.SEK  003423  FORMAT= 000001  LCDBN   003572  MS3     000032  OUT.34  001562
CURBLK  003614  FORTRK  010723  LDIACY  003574  MS30    002053  OVERFL= 000002
CURGRP  003620  FOR.SZ= 001375  LETTER  003625  MS31    002077  OVE.MN= 001364
CURPAT  003622  FRAME  = 000004  LINIT   002213  MS32    002130  OVE.MS= 000000
CURTRK  003621  FSTOP  = 100000  LINKLN= 000007  MS33    002156  OVL.MN= 010555
CVT   = 000020  FTLDEV= 040000  LOBYTE= 000377  MS34    002220  OVL.MS= 003673
CYLLO   003641  FTLSYS= 000000  LONG   = 003427  MS35    002245  OVL...= 014450
C2HARD= 040000  FT.BUF= 000000  LONGTO= 000001  MS36    002306  OVRLPT= 005774
DATPRE= 000005  FT.HI  = 000001  LOW    = 000002  MS37    002331  OVS.MN= 001040
DBNCYL= 000022  FT.LOW= 000002  LUNIT   002212  MS38    002422  OVS.MS= 022372
DCLOCK= 000004  FXBNCY  003601  MAXSND= 001750  MS4     000063  OV...  = 015070
DIA     003444  GCR     003452  MEDTYP= 000006  MS5     000110  PATPTR  003656
DINIT = 000011  GENEXT  011311  MESSAG= 060015  MS6     000135  PAT4    003663
DIS     003446  GENIN   011277  MICREV= 000003  MS7     000160  PAT5    003704
DISCON= 000204  GENOUT  011275  MOD     003464  MS8     000221  PAT6    003725
DMSDI   003542  GENPAT  011267  MODE    003465  MS9     000306  PAT8    003746
DONE  = 060016  GETCHR= 000207  MODE1   003466  MWR   = 000017  PHYSA = 001000
DONECD  002347  GETSTA= 000011  MODE2   003467  NOERR   012130  PORT2   002266
DRC     003450  GETSUB= 000210  MOD512= 126736  NSCSL   003543  PORT3   002322
DRTYPE= 000007  GETU    004377  MOD576= 074161  NUMPTR= 000014  PORT4   002332
DRVCLR= 000005  GROUP   003643  MRD    = 000016  OBUFF   003767  PORT5   002342
DRVID = 000004  GRPCNT  003631  MS.CLR  002611  OLDGRP  003632  RBNTRK= 000004
DRVONL= 000213  GRPCYL= 000002  MS.CNT= 152000  ONL     003442  RBUFD   004377
```

SYMBOL TABLE

```
RBUFLN= 000415        SDIVER= 000000        SNDONE= 003376        TRKBOT  011222        T7      007111
RBUFO   004370        SECCNT  003623        SS    = 000001        TRKEXT  011227        T7A     007124
RCONT = 000000        SECCYL  003617        SSCTOR  003645       TRKGRP= 000003        T7C     007165
RCTCPS= 000001        SECGRP  003616        ST      003474        TRKLOP  011175        T7X     007355
RCTCSZ= 000014        SECTRK  003603        STACK   002260        TRKTST  011162        T8      007440
RCV   = 000005        SEC512= 000400        START   004377        TSTBLK  003604        T8A     007511
RCVERR= 000400        SEC576= 000440        STATUS= 000007        TSTCMD  010566        T8B     007534
RCVRDY= 000001        SEEK    002621        STRST   002572        TSTCME  010717        T8D     007571
RDSTAT  001432        SEEKA   002634        STRTST  005774        TSTCM2  010570        T8E     007627
READB   011602        SEEK1   002652        STSPES= 000366        TSTCYL  003606        T8F     007715
READ1   003013        SEEK2   002653        ST.C  = 000002        T0A     006152        T8T     007616
READ1A  003104        SEEK3   003006        ST.CON= 000002        T0B     006162        T9      010411
READ1B  003247        SEND  = 000004        ST.DB = 001000        T08     007470        UDADM3= 001000 G
READ1C  003221        SERRTY  003473        ST.DF = 000020        T1MSIZ= 060000        UDA50 = 000000
READ1E  003251        SER00   002573        ST.DR = 000040        T10     010425        UDA52 = 000001
READ1X  003326        SER10   003052        ST.EL = 000010        T10LOP  010426        UNITNB  003355
REDERR  012033        SER11   003065        ST.ERR= 000002        T11     007717        UNITS   003335
REDEXT  012037        SER12   003103        ST.FE = 000200        T11ER   010606        UNIT0 = 000001
RETRY   003472        SER13   003115        ST.FO = 002000        T11X1   010645        UNIT1 = 000002
RETS  = 000001        SER14   003130        ST.MOD= 000001        T11X2   010672        UNIT2 = 000004
REVECT= 000004        SER15   003156        ST.MSK= 000000        T12     010542        UNIT3 = 000010
REVS  = 000007        SER16   003200        ST.OA = 000200        T2CMD = 060002        UNSSUC= 000175
RGO     011741        SER17   003223        ST.PE = 000040        T2DLL = 060001        UREAD = 000013
RM    = 000004        SER18   003255        ST.PS = 000002        T3A     006340        UTOTST= 060012
ROFDBN  003565        SER18A  003274        ST.RE = 000100        T3B     006372        UWRITE= 000014
ROFDC   003567        SER18B  003271        ST.RR = 000100        T3C     006254        U.SUBU= 000050
RREAL = 013400        SER18C  003266        ST.RTY= 000003        T3D     006373        U.UNUM= 000063
RSTOP = 100000        SER18D  003263        ST.RU = 000001        T3STRT  002265        VAR1    003633
RTDS    001371        SER18E  002261        ST.SR = 000020        T4A     006424        VAR2    003635
RTDSL   001422        SER22   003276        ST.STA= 000001        T4A'    006474        VAR3    003637
RUN     003456        SER23   003344        ST.S7 = 000400        T4B     006531        VAR4    003640
RUNCMD  003470        SER36   002446        ST.UNT= 000000        T4BB1 = 060005        WAITSI= 000012
RUNLBC= 000014        SER39   002532        ST.WE = 000010        T4BB2 = 060006        WBUFLN= 000401
RWRDY = 100000        SER40   003360        SUB   = 003516        T4C     006566        WCONT = 040000
RW.ANG= 000007        SER41   003402        SUBUNT  003454        T4D     006573        WFLAG   003627
RW.BUF= 000002        SER50   003415        T       006054        T4MPRM= 060003        WGO     011472
RW.CMD= 000005        SER51   003422        TALK    001563        T4MXFR= 060011        WREAL = 122400
RW.CPT= 000000        SER52   003430        TESTED  002400        T4SEEK= 060010        WRITEB  011314
RW.HI = 000004        SER53   003467        TESTEV  002364        T4SOFT= 060007        WRONG = 000002
RW.LOW= 000003        SER54   003520        TESTEW  002355        T4UPRM= 060004        WRTEXT  011600
RW.SDI= 000006        SER55   003551        TESTEX  002372        T5      010067        WSTOP = 140000
RW.STA= 000001        SER56   003604        TESTEY  002376        T6      006606        WTRCMP  011231
SBCRES= 000167        SER57   003634        TESTX   002336        T6A     006615        WTREXT  011265
SCR     003453        SHRTTO= 000000        TEST4 = 000000        T6C     006660        XBNCYL= 000021
SCTWRD= 000377        SL.GRP= 107000        TIMEOU= 000001        T6D     006665        XFERRT= 000000
SDI     003356        SNDAGN  001473        TO      002175        T6E     007072        XREAD = 000002
SDILTO  002174        SNDLV1  010552        TOOBIG= 000001        T6F     007111        XWRITE= 000003
SDISTO  002173        SNDL1A  010555        TRACK   003644        T6G     007040

. ABS.  032160      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  21042 WORDS  ( 83 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  70 PAGES
B:UDAT3.052,B:UDAT3.L52/C=[30,30]DMAC52/M,B.UDAT3
```

CROSS REFERENCE TABLE (CREF V04.00)

```
MS32     59-35    66-43#
MS33     54-94    66-43#
MS34     66-43#
MS35     53-19    66-43#
MS36     57-117   66-43#
MS37     57-119   66-43#
MS38     57-111   66-43#
MS4      24-51    66-43#
MS5      24-55    66-43#
MS6      24-59    66-43#
MS7      24-64    66-43#
MS8      24-71    66-43#
MS9      42-37    66-43#
MWR       4-18#   22-22
NOERR    66-12    66-20#
NSCSL    34-70#   48-12*
NUMPTR   21-8#    24-23#   24-43#   24-47#   24-51#   24-55#   24-59#   24-60    24-60    24-60    24-60    24-60#   24-60#   24-60#
         24-64#   24-65    24-65    24-65    24-65    24-65    24-65    24-65#   24-65#   24-65#   24-65#   24-71#   24-72    24-72
         24-72    24-72    24-72    24-72    24-72    24-72    24-72#   24-72#   24-72#   24-72#   24-72#   29-2     29-2     29-2
         29-2     29-2#    29-2#    29-2#    32-29#   32-33#   32-41    32-41    32-41    32-41    32-41    32-41    32-41#   32-41#
         32-41#   32-41#   33-60#   33-64#   33-70#   33-96    33-96    33-96    33-96    33-96    33-96    33-96    33-96    33-96
         33-96    33-96    33-96    33-96    33-96    33-96#   33-96#   33-96#   33-96#   33-96#   33-96#   33-96#   33-96#   39-38#
         42-37#   46-29    46-29    46-29#   46-29#   46-38    46-38    46-38#   46-38#   46-44#   48-37#   49-46#   49-53#   53-19#
         54-94#   57-111   57-111   57-111#  57-111#  57-117   57-117   57-117#  57-117#  57-119#  58-17#   58-20#   58-28    58-28
         58-28#   58-28#   59-31    59-31    59-31    59-31    59-31#   59-31#   59-31#   59-33    59-33    59-33#   59-33#   59-35
         59-35    59-35#   59-35#   60-35#   60-38#   60-43#   60-67    60-67    60-67    60-67    60-67    60-67    60-67    60-67
         60-67#   60-67#   60-67#   60-67#   60-67#   64-38#   64-41#   64-48#   64-71    64-71    64-71    64-71    64-71    64-71
         64-71    64-71    64-71    64-71    64-71    64-71    64-71    64-71    64-71#   64-71#   64-71#   64-71#   64-71#   64-71#
         64-71#   64-71#   65-26#   65-29#   65-36#   65-55    65-55    65-55    65-55    65-55    65-55    65-55    65-55    65-55
         65-55    65-55    65-55    65-55    65-55    65-55#   65-55#   65-55#   65-55#   65-55#   65-55#   65-55#   65-55#   66-15
         66-15    66-15    66-15    66-15    66-15    66-15    66-15    66-15    66-15    66-15    66-15    66-15    66-15    66-15#
         66-15#   66-15#   66-15#   66-15#   66-15#   66-15#   66-15#
OBUFF    35-80#   60-16    63-12    64-30    66-7
OLDGRP   30-52*   30-57*   30-62*   34-109#  50-30*   50-50*
ONL      34-18    34-33#
OUT.01   21-8*    23-6#    24-23*   24-43*   24-47*   24-51*   24-55*   24-59*   24-64*   24-71*   32-29*   32-33*   32-41*   33-60*
         33-64*   33-70*   33-96*   38-6     39-38*   42-37*   46-29*   46-38*   46-44*   48-37*   49-46*   49-53*   50-75*   53-19*
         54-94*   57-111*  57-117*  57-119*  58-17*   58-20*   58-28*   59-31*   59-33*   59-35*   60-35*   60-38*   60-43*   60-67*
         64-38*   64-41*   64-48*   64-71*   65-26*   65-29*   65-36*   65-55*   66-15*
OUT.02   21-8*    23-7#    24-23*   24-43*   24-47*   24-51*   24-55*   24-59*   24-64*   24-71*   32-29*   32-33*   32-41*   33-60*
         33-64*   33-70*   33-96*   39-38*   42-37*   46-29*   46-38*   46-44*   48-37*   49-46*   49-53*   50-76*   53-19*   54-94*
         57-111*  57-117*  57-119*  58-17*   58-20*   58-28*   59-31*   59-33*   59-35*   60-35*   60-38*   60-43*   60-67*   64-38*
         64-41*   64-48*   64-71*   65-26*   65-29*   65-36*   65-55*   66-15*
OUT.03   21-8*    22-13*   23-8#    24-23*   24-43*   24-47*   24-51*   24-55*   24-59*   24-64*   24-71*   32-29*   32-33*   32-41*
         33-60*   33-64*   33-70*   33-96*   39-5*    39-38*   42-37*   46-29*   46-38*   46-44*   48-37*   49-46*   49-53*   50-77*
         53-19*   54-94*   57-111*  57-117*  57-119*  58-17*   58-20*   58-28*   59-31*   59-33*   59-35*   60-35*   60-38*   60-43*
         60-67*   64-38*   64-41*   64-48*   64-71*   65-26*   65-29*   65-36*   65-55*   66-15*
OUT.04   21-8*    23-9#    24-23*   24-43*   24-47*   24-51*   24-55*   24-59*   24-64*   24-71*   32-29*   32-33*   32-41*   33-60*
         33-64*   33-70*   33-96*   39-38*   42-37*   46-29*   46-36    46-44*   48-37*   49-46*   49-53*   50-78*   53-19*   54-94*
         57-111*  57-117*  57-119*  58-17*   58-20*   58-28*   59-31*   59-33*   59-35*   60-35*   60-38*   60-43*   60-67*   64-38*
         64-41*   64-48*   64-71*   65-26*   65-29*   65-36*   65-55*   66-15*
OUT.05   23-10#   24-60*   24-65*   24-72*   29-2*    29-4*    32-41*   33-96*   39-6     46-29*   50-79*   57-111*  57-117*  58-28*
         59-31*   59-33*   59-35*   60-67*   64-71*   65-37*   65-55*   66-15*
OUT.06   23-11#   24-60*   24-65*   24-72*   29-2*    32-41*   33-96*   46-31    59-31*   60-67*   64-71*   65-55*   66-15*
OUT.07   23-12#   24-61*   24-65*   24-72*   32-41*   33-96*   60-67*   64-71*   65-55*   66-15*
OUT.08   23-13#   24-66*   24-72*   32-44*   33-96*   60-67*   64-71*   65-55*   66-15*
```

```
OUT.09  23-14#  24-73*  33-96*  64-71*  65-55*  66-15*
OUT.10  23-15#  33-96*  64-71*  65-55*  66-15*
OUT.11  23-16#  33-96*  64-71*  65-55*  66-15*
OUT.12  23-17#  33-97*  64-72*  65-56*
OUT.13  23-18#
OUT.14  23-19#
OUT.15  23-20#
OUT.16  23-21#
OUT.17  23-22#
OUT.18  23-23#
OUT.19  23-24#  60-68*
OUT.20  23-25#
OUT.21  23-26#
OUT.22  23-27#
OUT.23  23-28#
OUT.24  23-29#
OUT.25  23-30#
OUT.26  23-31#
OUT.27  23-32#
OUT.28  23-33#
OUT.29  23-34#
OUT.30  23-35#
OUT.31  23-36#
OUT.32  23-37#
OUT.33  23-38#
OUT.34  23-39#
OUT.RQ  21-8*   22-14*  22-15   22-20   22-23*  23-5#   23-40   24-23*  24-43*  24-47*  24-51*  24-55*  24-59*  24-64*
        24-71*  29-5    29-10   32-29*  32-33*  32-41*  33-60*  33-64*  33-70*  33-96*  39-38*  39-39   42-37*  46-29*
        46-44*  48-37*  49-46*  49-53*  53-19*  54-94*  57-111* 57-117* 57-119* 58-17*  58-20*  58-28*  59-31*  59-33*
        59-35*  60-35*  60-38*  60-43*  60-67*  64-38*  64-41*  64-48*  64-71*  65-26*  65-29*  65-36*  65-55*  66-15*
        66-16
OV...    2-9     2-9     2-9#   66-43   66-43   66-43   66-43#  66-47   66-47#
OVE.MN   2-9#   66-43
OVE.MS  66-43#  66-47
OVERFL   5-75#
OVL...   2-9     2-9#   66-43   66-43#  66-47   66-47#
OVL.MN   2-9     2-9    66-43   66-43   66-43#
OVL.MS  66-43#  66-47   66-47   66-47#
OVRLPT  39-56#
OVS.MN   2-9#
OVS.MS  66-43#
PAT4    35-3    35-10#
PAT5    35-4    35-27#
PAT6    35-5    35-44#
PAT8    35-6    35-61#
PATPTR  35-2#   62-9    62-22
PHYSA    2-9     2-9     2-9     2-9     2-9#
PORT2   27-11#  27-52
PORT3   27-27#  27-31
PORT4   27-28   27-32#
PORT5   27-16   27-18   27-20   27-50#
RBNTRK   5-31#  48-5
RBUF0   33-16   33-53   33-87   33-96   33-96   33-96   35-85#
RBUFD   35-94   36-7#   57-59   57-113* 57-115* 65-18   66-8
RBUFLN   3-99#
RCONT    3-64#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
RCTCPS    5-8#    43-19
RCTCSZ    5-40#
RCV       4-8#    24-34    36-100
RCVERR    6-25#   21-32    32-27    33-58    58-15    60-33    64-36    65-24
RCVRDY    6-21#   36-68    42-33
RDSTAT   21-15    21-20#   32-24    33-55    36-59    36-67    37-13    58-12    60-30    64-33    65-21
READ1    33-14#   49-95    50-61    50-69
READ1A   33-53#   33-95
READ1B   33-91    33-95#
READ1C   33-69    33-73#
READ1E   33-86    33-96#
READ1X   33-62    33-66    33-72    33-84    33-101#
READB    62-14    65-2#
REDERR   65-38    65-57#
REDEXT   65-35    65-52    65-54    65-61#
RETRY    34-60#   57-7*    57-102*  57-103*
RETS      5-7#
REVECT    5-63#
REVS      5-16#
RGO      65-33    65-39#
RM        5-32#
ROFDBN   34-80#   49-76*   49-77*
ROFDC    34-81#   49-48*   49-49*   49-60*   49-62*   49-68*   54-60*   54-62*   54-64*
RREAL     3-67#   35-96    57-58    65-16
RSTOP     3-63#   35-89    35-93    57-43    65-6
RTDS     21-5#    42-27    44-10    46-13
RTDSL    21-5     21-13#   57-24
RUN      34-29    34-45#   54-81*   55-5*    55-13*   55-15*   56-5*    56-8*    56-10*   56-14*   56-19*   56-22*   56-26*   56-27*
         56-29*   58-33*   59-4*    59-29*   59-33
RUNCMD   34-56#   55-15    58-33    59-29
RUNLBC   34-57#   56-27    59-4
RW.ANG    3-57#
RW.BUF    3-52#    3-53    57-60*   64-31*   65-19*
RW.CMD    3-55#    3-56    33-96    57-57*   57-58*   64-29*   65-17*
RW.CPT    3-42#    3-43    57-43*   57-88*   64-10*   65-10*
RW.HI     3-54#    3-55    33-96    57-56*   64-26*
RW.LOW    3-53#    3-54    33-16    33-87    33-96    57-47*   64-16*
RW.SDI    3-56#    3-57
RW.STA    3-43#    3-52    64-14*   65-14*
RWRDY     6-26#   32-37    33-68    46-40    57-29    60-41    64-44    65-32
SBCRES    6-44#
SCR      34-23    34-42#
SCTWRD    7-8#
SDI      21-14    24-8     25-19    27-32*   29-8     31-9     32-11    32-23    33-54    33-73    34-13#   42-14    54-84    57-15
         57-61    58-11    59-6     60-29    64-32    64-51    65-20    65-39
SDILTO   24-30    24-78#   43-18*   54-75*   57-22
SDISTO   24-28    24-77#   43-4*    43-14*
SDIVER    5-4#
SEC512   57-69#   57-71
SEC576   57-70#
SECCNT   34-99#   61-13*   61-25    64-46    64-69    65-34    65-53    66-13
SECCYL   34-95#   49-19*   49-20*   50-49
SECGRP   34-94#   49-14*   49-20    49-35    49-71    50-48
SECTRK   30-17    30-31    34-86#   48-10*   49-10    49-94    49-97    50-47    50-60    50-66    54-41    57-12    60-17    61-11
SEEK     32-10#   47-33    49-92    50-58    50-67    53-13    54-31    54-67    57-21
SEEK1    32-19    32-22#
```

CROSS REFERENCE TABLE (CREF V04.00)

TABLE OF CONTENTS

```
                                .TITLE  UDAT3   DISK FUNCTIONAL
                        ;
                        ;
                        ; COPYRIGHT (C)  1981
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
000001                  UDA50=1
000000                  UDA52=0
                        ;THIS PROGRAM SHALL BE ASSEMBLED WITH THE PROGRAM DMACRO
                        ;USING A COMMAND LINE SIMILAR TO:
                        ;
                        ;  UDAT3,UDAT3/C=[1,2]DMACRO,UDAT3
                        ;
                        ;THEN LINK THE OBJECT FILE USING A COMMAND LINE SIMILAR TO:
                        ;
                        ;  UDAT3.BIN/L=UDAT3
                        ;
                        ;
000000                  TEST4   =       0                       ; THIS IS NOT TEST4
```

```
      1                                       .SBTTL  START OF TEST CODE
     14 000000                                DMCODE  UDADM3,0,714,3,0,1000
     18
     27 000714  114007                        CLR R0                          ;CHANGE TO BREAKPOINT FOR DEBUG
     31
     32                                 ;INITIALIZE STACK
     33
     34 000715  104206  001542                MOV #STACK,SP                   ;SET UP STACK POINTER
     44 000717  003665                        BR      START                   ; BRANCH OVER SUPPORT CODE
     45
     46
```

```
                                .SBTTL  UDA DM PROGRAM PARAMETERS
                                .LIST MEB
                        ;
                        ;       EQUATES
                        ;
                        ;       HIGHEST USABLE LOCATION OF UDA MEMORY + 1
                        ;
007774                  HIMEM   =       7774                    ; HIGH MEMORY+1
007774                  OVSTRT  =       7774                    ; OVERLAY ADDRESS LOCATION
                        ;
                        ;       OFFSETS FOR FORMAT TRACK TABLE
                        ;
000000                  FT.BUF  =       0.                      ;BUFFER POINTER OFFSET
000001                  FT.HI   =       1.                      ;HI ORDER HEADER OFFSET
000002                  FT.LOW  =       2.                      ;LOW ORDER HEADER OFFSET
                        ;
                        ;       OFFSETS FOR FORMAT TRACK BUFFER
                        ;
000000                  FB.DAT  =         0.                    ;FIRST DATA WORD OFFSET
000400                  FB.EDC  =       256.                    ;EDC WORD OFFSET
                        ;
                        ;       OFFSETS FOR READ/WRITE I/O CHAIN TABLES
                        ;
000000                  RW.STAT =       0.                      ;STATUS AND NEXT BUFFER POINTER OFFSET
000001                  RW.BUF  =       RW.STAT+1.              ;POINTER TO DATA BUFFER
000002                  RW.LOW  =       RW.BUF+1.               ;HI ORDER EXPECTED HEADER
000003                  RW.HI   =       RW.LOW+1.               ;LOW ORDER EXPECTED HEADER
000004                  RW.CMD  =       RW.HI+1.                ;SDI COMMAND AND HEAD ADDRESS
000005                  RW.SDI  =       RW.CMD+1.               ;DUMMY SDI CONTROL BLOCK POINTER
000006                  RW.ANG  =       RW.SDI+1.               ;THETA FROM INDEX
                        ;
                        ;       CONSTANTS FOR READ AND WRITE XFC'S
                        ;
140000                  WSTOP   =       140000                  ; LAST ENTRY IN CHAIN FOR WRITE
040000                  WCONT   =       40000                   ; WRITE CONTINUE
100000                  RSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR READ
000000                  RCONT   =       0                       ; READ CONTINUE
100000                  FSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR FORMAT
122400                  WREAL   =       122400                  ; WRITE REAL TIME ECOMMAND
013400                  RREAL   =       13400                   ; READ REAL TIME COMMAND
010000                  ECCFLG  =       10000                   ; ECC ERROR IN BUFFER BIT
100000                  EOC     =       100000                  ; END OF CHAIN
040000                  BUFFLG  =       40000                   ; BUFFER FULL OR EMPTY FLAG
001750                  MAXSND  =       1000.
                        ;
                        ;       HEADER CODES
                        ;
000000                  HD.LBN  =       000000                  ;GOOD LBN
060000                  HD.RBN  =       060000                  ;GOOD RBN, PERHAPS UNUSED
030000                  HD.REV  =       030000                  ;REVECTORED LBN
110000                  HD.BAD  =       110000                  ;BAD BLOCK
050000                  HD.PRV  =       050000                  ;PRIMARY REVECTORED BLOCK
```

```
     81          120000                  HD.XBN  =       120000                  ;XBN BLOCK
     82          140000                  HD.DBN  =       140000                  ;DBN BLOCK
     83
     84                                  ;       LEVEL 1 CODES
     85          070400                  MS.STR  =       70400                   ;MESSAGE START
     86          131000                  MS.END  =       131000                  ;MESSAGE END
     87          152000                  MS.CNT  =       152000                  ;MESSAGE CONTINUE
     88          107000                  SL.GRP  =       107000                  ;SELECT GROUP
     89
     90                                  ;       OFFSETS FOR DATA BUFFERS
     91
     92          000000                  BF.DAT  =         0.                    ;DATA
     93          000400                  BF.EDC  =       256.                    ;ERROR DETECTION CODE
     94          000401                  BF.ECC  =       257.                    ;LAST 17 ECC RESIDUES
     95
     96                                  ;       BUFFER AND READ/WRITE CHAIN LINK SIZES
     97
     98          000401                  WBUFLN  =       257.                    ; WRITE BUFFER SIZE
     99          000415                  RBUFLN  =       WBUFLN+12.              ; READ BUFFER SIZE
    100          000007                  LINKLN  =       7.                      ; LINK SIZE
    101
    102                                  ;       UNIT PARAMETER USED BY TEST3
    103
    104          000050                  U.SUBU  =       50
    105          000063                  U.UNUM  =       63
    106
    107                                  ;       FORMAT CHAIN SIZE
    108
    109          001375                  FOR.SZ  =       255.*3
```

```
                        ;       XFC DEFINITION EQUATES
                        ;
000000                  BREAK   =       0.              ;BREAKPOINT XFC CODE
000001                  FORMAT  =       1.              ;FORMAT TRACK XFC CODE
000002                  XREAD   =       2.              ;READ N SECTORS XFC CODE
000003                  XWRITE  =       3.              ;WRITE N SECTORS XFC CODE
000004                  SEND    =       4.              ;SEND SDI COMMAND XFC CODE
000005                  RCV     =       5.              ;RECEIVE SDI MESSAGE XFC CODE
000006                  COMPARE =       6.              ;COMPARE DATA PATTERN TO BUFFER
000007                  STATUS  =       7.              ;RETURN DRIVE STATUS XFC CODE
000010                  ECHO    =       8.              ;ECHO DATA TO DRIVE XFC CODE
000011                  DINIT   =       9.              ;DRIVE INITIALIZE XFC CODE
000012                  WAITSI  =       10.             ;WAIT FOR SECTOR OR INDEX PULSE
000013                  UREAD   =       11.             ;READ UNIBUS MEMORY XFC CODE
000014                  UWRITE  =       12.             ;WRITE UNIBUS MEMORY XFC CODE
000015                  ECC     =       13.             ;DO ECC ON BUFFER XFC CODE
000016                  MRD     =       14.             ;SEND BUFFER TO MAINTENANCE READ COMMAND
000017                  MWR     =       15.             ;GET BUFFER FROM MAINTENANCE WRITE COMMAND
000020                  CVT     =       16.             ;CONVERT TO PHYSICAL ADDRESS XFC CODE
000021                  EXIT    =       17.             ;TERMINATE DM PROGRAM XFC CODE
                        ;
                        ;       MEDIA TYPE
                        ;
126736                  MOD512 = 126736
074161                  MOD576 = 074161
                        ;
                        ;       GET STATUS OFFSETS
                        ;
000000                  ST.UNT  =       0.              ;UNIT NUMBER
000000                  ST.MSK  =       0.              ;SUBUNIT MASK
000001                  ST.STA  =       1.              ;STATUS BYTE
000001                  ST.MOD  =       1.              ;MODE BYTE
000002                  ST.ERR  =       2.              ;ERROR BYTE
000002                  ST.CON  =       2.              ;CONTROLLER BYTE
000002                  ST.C    =       2.              ;C BITS
000003                  ST.RTY  =       3.              ;RETRY COUNT/FAILURE CODE
                        ;
                        ;       STATUS BIT DEFINITIONS
                        ;
000010                  ST.EL   =       10              ; LOGGABLE INFO IN EXTENDED STATUS
000200                  ST.OA   =       200             ; ONLINE TO ANOTHER (SET IF DRIVE UNAVAILABLE)
000100                  ST.RR   =       100             ; READJUSTMENT BIT (SET IF RECALIBRATION REQUIRED)
000040                  ST.DR   =       40              ; DIAGNOSTIC REQUEST (SET IF DIAGNOSTIC REQUESTED)
000020                  ST.SR   =       20              ; SPINDLE READY (SET IF SPINDLE READY)
000002                  ST.PS   =       2               ; PORT SWITCH (SET IF PORT SWITCH IN)
000001                  ST.RU   =       1               ; RUN/STOP SWITCH (SET IF RUN/STOP SWITCH IN)
000200                  ST.FE   =       200             ; FATAL ERROR (SET IF FATAL ERROR OCCURRED)
000100                  ST.RE   =       100             ; RETRIABLE ERROR (SET IF RETRIABLE ERROR OCCURRED)
000040                  ST.PE   =       40              ; PROTOCOL ERROR (SET IF PROTOCOL ERROR OCCURRED)
000020                  ST.DF   =       20              ; INITIALIZATION FAILURE (SET IF INIT FAILED)
000010                  ST.WE   =       10              ; WRITE ENABLE (SET IF WRT ATTEMPTED ON PROT DISK)
002000                  ST.FO   =       2000            ; FORMATTING (SET IF FORMATTING ENABLED)
001000                  ST.DB   =       1000            ; DIAGNOSTIC CYLS (SET IF DIAG CYL ACCESS ENABLED)
000400                  ST.S7   =       400             ; SECTOR SIZE (SET FOR 576 BYTE SECTORS)
```

```
                        ;       GET COMMON CHARACTERISTICS OFFSETS
                        ;
000000          SHRTTO  =       0.              ;SHORT TIMEOUT <3:0>
000000          SDIVER  =       0.              ;SDI VERSION <7:4>
000000          XFERRT  =       0.              ;TRANSFER RATE <15:0>
000001          LONGTO  =       1.              ;LONG TIMEOUT <3:0>
000001          RETS    =       1.              ;RETRIES <7:4>
000001          RCTCPS  =       1.              ;F/RCT COPIES <11:8>
000001          SS      =       1.              ;SECTOR SIZE <15:15>
000002          ERLEV   =       2.              ;ERROR RETRY LEVELS <7:0>
000002          ECCRSH  =       2.              ;ECC THRESHOLD <15:8>
000003          MICREV  =       3.              ;MICROCODE REVISION NUMBER <7:0>
000003          HRDREV  =       3.              ;HARDWARE REVISION NUMBER <15:8>
000004          DRVID   =       4.              ;UNIQUE DRIVE ID <47:0>
000007          DRTYPE  =       7.              ;DRIVE TYPE IDENTIFIER <7:0>
000007          REVS    =       7.              ;REVS/SECOND <15:8>
                        ;
                        ;       GET SUBUNIT CHARACTERISTICS OFFSETS
                        ;
                        ;THESE OFFSETS ARE CURRENTLY GIVEN AS FOLLOWING THE COMMON CHARACTERISTICS
                        ;
000013          SUB     =       11.             ;OFFSET TO PUT SUBUNIT AFTER COMMON;
000000          LBNCYL  =       0.              ;NUMBER OF CYLINDERS IN LBN AREA <31:0>
000001          HICYL   =       1.              ;HI ORDER CYLINDER BITS <15:12>
000002          GRPCYL  =       2.              ;GROUPS PER CYLINDER <7:0>
000002          HILBN   =       2.              ;HI STARTING LBN <11:8>
000002          HIXBN   =       2.              ;HI STARTING XBN <15:12>
000003          TRKGRP  =       3.              ;TRACKS PER GROUP <7:0>
000003          HIRBN   =       3.              ;HI STARTING RBN <11:8>
000003          HIDBN   =       3.              ;HI STARTING DBN <15:12>
000004          RBNTRK  =       4.              ;RBNS PER TRACK <6:0>
000004          RM      =       4.              ;REMOVABLE MEDIA <7:7> 1=REMOVEABLE
000005          DATPRE  =       5.              ;DATA PREAMBLE SIZE IN WORDS <7:0>
000005          HDRPRE  =       5.              ;HEADER PREAMBLE SIZE IN WORDS <15:8>
000006          MEDTYP  =       6.              ;MEDIA TYPE <31:0>
000010          FCTSIZ  =       8.              ;FCT COPY SIZE <15:0>
000011          LBNTRK  =       9.              ;LBNS PER TRACK <7:0>
000011          GRPOFF  =       9.              ;GROUP OFFSET (SECTORS) <15:8>
000012          LBNHST  =       10.             ;LBNS IN HOST AREA <31:0>
000014          RCTCSZ  =       12.             ;RCT COPY SIZE <15:0>
000021          XBNCYL  =       17.             ;CYLS IN XBN AREA <15:0>
000022          DBNCYL  =       18.             ;CYLS IN DBN AREA <15:8>
                        ;
                        ;       UNIT CODES
                        ;
000001          UNIT0   =       1.              ;UNIT ZERO CODE
000002          UNIT1   =       2.              ;UNIT ONE CODE
000004          UNIT2   =       4.              ;UNIT TWO CODE
000010          UNIT3   =       8.              ;UNIT THREE CODE
                        ;
                        ;       BIT MASK DEFINITIONS
                        ;
                        ;
177400          HIBYTE  =       177400          ;HIGH BYTE MASK
000377          LOBYTE  =       000377          ;LOW BYTE MASK
007777          HBHINB  =       7777            ;HI BYTE, HI NIBBLE MASK
170377          HBLONB  =       170377          ;HI BYTE, LO NIBBLE MASK
```

```
                177417          LBHINB  =       177417          ;LO BYTE, HI NIBBLE MASK
                177760          LBLONB  =       177760          ;LO BYTE, LO NIBBLE MASK
                                ;
                000001          TIMEOUT =       1.              ;DRIVE TIMEOUT CODE
                000002          HEADER  =       2.              ;HEADER COMPARE FAILURE CODE
                000004          REVECT  =       4.              ;REVECTOR NEEDED CODE
                                ;
                000002          WRONG   =       2.              ;FIRST WORD NOT START FRAME CODE
                000004          FRAME   =       4.              ;FRAMING ERROR CODE
                000010          CHECK   =       8.              ;CHECKSUM ERROR CODE
                                ;
                000001          TOOBIG  =       1.              ;NUMBER OF WORDS EXCEEDS 7064
                000002          LOW     =       2.              ;DM BUFFER ADDRESS IS LESS THAN 714
                                ;
                                ;       BIT MASK
                                ;
                000001          LARGE   =       1.              ;BLOCK NUMBER TOO LARGE
                000002          OVERFL  =       2.              ;SECTOR NUMBER LARGER THAN 16 BITS
```

```
                                ;MAINTANENCE READ/WRITE REQUEST NUMBERS
                                ;
060000                          T1MSIZ  =       0.+60000        ;GET FREE MEMORY PARAMETERS
060001                          T2DLL   =       1.+60000        ;DOWNLINE LOAD DRIVE DIAGNOSTIC
060002                          T2CMD   =       2.+60000        ;MANUAL INTERVENTION TEST 2 PROTOCOL
060003                          T4MPRM  =       3.+60000        ;GET MASTER PARAMETERS FROM SW QUESTIONS
060004                          T4UPRM  =       4.+60000        ;GET UNIT PARAMETERS FROM HW QUESTIONS
060005                          T4BB1   =       5.+60000        ;GET BAD BLOCKS (1 THRU 14)
060006                          T4BB2   =       6.+60000        ;GET REST OF BAD BLOCKS (15 AND 16)
060007                          T4SOFT  =       7.+60000        ;ADD TO SOFT ERROR AND ECC COUNT
060010                          T4SEEK  =       8.+60000        ;ADD 1 TO SEEK COUNT
060011                          T4MXFR  =       9.+60000        ;ADD TO MEGABITS READ AND WRITTEN
060012                          UTOTST  =       10.+60000       ;GET UNITS TO TEST
060013                          ERRMES  =       11.+60000       ;PRINT ERROR MESSAGE
060014                          ERRMC   =       12.+60000       ;TEST 4 ERROR REPORTING
060015                          MESSAG  =       13.+60000       ;INFORMATION MESSAGE
060016                          DONE    =       14.+60000       ;MARK DM PROGRAM AS NO LONGER RUNNING
                                ;
                                ;       OTHER BIT DEFINITIIONS
                                ;
000001                          RCVRDY  =       1               ; RECIEVER READY 1 = READY
000002                          ATTN    =       2               ; ATTENTION BIT FOR RETURN DRIVE SIGNALS XFC
000004                          DCLOCK  =       4               ; TRANSMIT ERROR
000100                          AVAIL   =       100             ; AVAILABLE 1 = AVAILABLE
000400                          RCVERR  =       400             ; RECIEVER ERROR
100000                          RWRDY   =       100000          ; IF SET, UDA IS ABLE TO READ AND/OR WRITE TO DRIVE
                                ;
                                ;       SDI COMMANDS AND RESPONSES
                                ;
000204                          DISCON  =       204             ; DISCONNECT DRIVE
000006                          ERECOV  =       6               ; ERROR RECOVERY
000201                          CHGMOD  =       201             ; CHANGE MODE
000213                          DRVONL  =       213             ; DRIVE ONLINE
000014                          DRVRUN  =       14              ; DRIVE RUN
000005                          DRVCLR  =       5               ; DRIVE CLEAR OPCODE
000207                          GETCHR  =       207             ; GET CHARACTERISTICS
000210                          GETSUB  =       210             ; GET SUBUNIT CHARACTERISTICS
000011                          GETSTA  =       11              ; GET STATUS
000216                          IRECLB  =       216             ; RECALIBRATE
000012                          INSEEK  =       12              ; INITIATE SEEK
000176                          COMPLT  =       176             ; SUCCESSFUL COMPLETION
000175                          UNSSUC  =       175             ; UNSUCCESSFUL COMPLETION
000170                          CHRRES  =       170             ; GET CHARACTERISTICS RESPONSE
000167                          SBCRES  =       167             ; GET SUBUNIT CHARACTERISTICS RESPONSE
000366                          STSRES  =       366             ; GET STATUS RESPONSE
000350                          ECHOC   =       350             ; DIAGNOSTIC ECHO COMMAND AND RESPONSE
                                ;
                                ;       ERROR CODES
                                ;
000000                          FTLSYS  =               0       ; SYSTEM FATAL ERROR
040000                          FTLDEV  =           40000       ; DEVICE FATAL
100000                          ERHARD  =          100000       ; HARD ERROR
140000                          ERSOFT  =          140000       ; SOFT ERROR
040000                          C2HARD  =       <ERHARD&^CERSOFT>!<ERSOFT&^CERHARD>     ; CHANGE SOFT TO HARD ERROR
```

```
                                        .SBTTL  TEST 4 SPECIFIC INFORMATION
                                ;
                                ;       TEST 4 SPECIFIC INFORMATION
                                ;
                                ;
                                ;       CONSTANTS
                                ;
000377                  SCTWRD  =       255.                    ; NUMBER OF WORDS IN SECTOR TO FILL
000105                  INTEDC  =       69.                     ; INITIAL EDC VALUE
```

```
                        .SBTTL  MACRO DEFINITIONS
                ;
                ;       MESSAGE CONTROL TABLE MACRO
                ;
                        .MACRO MSG CMDBUF,CMDSZ,RPLBUF,RPLSZ,SUCCOM
                        .WORD CMDBUF                    ;ADDRESS OF COMMAND
                        .WORD CMDSZ                     ;SIZE OF COMMAND IN BYTES
                        .WORD RPLBUF                    ;ADDRESS OF REPLY
                        .WORD RPLSZ                     ;SIZE OF REPLY IN WORDS
                        .IF NB NUMBER
                        .WORD SUCCOM                    ; SUCCESSFUL COMPLETION CODE
                        .ENDC
                        .ENDM
```

MACRO DEFINITIONS

```
      1                                 .MACRO  BCS     LAB..,?B
      2                                                                 BCC     B
      3                                                                 BR      LAB..
      4                           B:
      5                                 .ENDM
```

MACRO DEFINITIONS

```
                        ;       PUSH REGISTER MACRO
                        ;
                                .MACRO PUSH R9
                                .IRP X,<R9>
                                                        MOV X,-(SP)
                                .ENDR
                                .ENDM
                        ;
                        ;       POP REGISTER MACRO
                        ;
                                .MACRO POP R9
                                .IRP X,<R9>
                                                        MOV (SP)+,X
                                .ENDR
                                .ENDM
```

```
                        ;ERROR MACROS
                        ;THESE MACROS ARE CALLED TO REPORT ERRORS TO THE HOST PROGRAM.
                        ;THE MACRO NAMES ARE : ERRSF, ERRDF, ERRHRD, ERRSFT. EACH RESULTS IN THE HOST
                        ;BEING REQUESTED TO REPORT THE ERROR.
                        ;ARGUMENTS:     1 (MS$) MESSAGE POINTER
                        ;               2 (P1$) PARAMETER #1
                        ;               3 (P2$) PARAMETER #2
                        ;               4 (P3$) PARAMETER #3
                        ;               5 (P4$) PARAMETER #4
                        ;               6 (P5$) PARAMETER #5
                        ;               7 (P6$) PARAMETER #6
                        ;               8 (P7$) PARAMETER #7
                        ;               9 (P8$) PARAMETER #8
                        ;
                        ;THE MESSAGE POINTER MUST POINT TO AN ADDRESS IN THE OVERLAY 'MS' IMMEDIATELY
                        ;FOLLOWING THE MAIN CODE. ANY ADDRESS MODE MAY BE USED (E.G. #MS1, @R2).
                        ;THE ADDRESS MUST CONTAIN AN ASCII FORMAT STRING TO DETERMINE THE MESSAGE
                        ;TO PRINT.
                        ;THE PARAMETER ARGUMENTS ARE OPTIONAL. THEY SHOULD BE SUPPLIED ONLY WHEN
                        ;THERE IS DATA TO BE PASSED TO THE HOST THAT WILL BE USED IN PRINTING THE
                        ;MESSAGE. THESE PARAMETER ARGUMENTS ARE THE ADDRESS OF DATA TO BE PASSED
                        ;USING ANY ADDRESSING MODE DESIRED.
                        ;ALL REGISTERS ARE RETURNED UNCHANGED. IT SHOULD BE NOTED THAT ARGUMENTS
                        ;CONTAINING SOMETHING OTHER THAN A REGISTER NAME (E.G. #100 OR MEMADR)
                        ;ASSEMBLE TO INSTRUCTIONS THAT SAVE AND RESTORE A REGISTER ON THE STACK.

                                .MACRO ERRSF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ FTLSYS,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRDF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ FTLDEV,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRHRD MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NLIST
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ ERHARD,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .LIST
                                .ENDM

                                .MACRO ERRSFT MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ ERSOFT,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM
```

```
                        ;THE FOLLOWING MACRO ACTUALLY PROCESSES THE ERROR CALL TO THE HOST PROGRAM

                                .MACRO ERROR$ ET$,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,ERRN$
                                  .RADIX 8
                                  PRMS=ARGS$-1
                                  .IF LT,<PRMS>
                                    .ERROR;NOT ENOUGH ARGUMENTS IN ERROR CALL
                                  .ENDC
                                  REGS$=-1
                                  .IIF GE,<PRMS-8.>,PARGS. P8$
                                  .IIF GE,<PRMS-7.>,PARGS. P7$
                                  .IIF GE,<PRMS-6.>,PARGS. P6$
                                  .IIF GE,<PRMS-5.>,PARGS. P5$
                                  .IIF GE,<PRMS-4.>,PARGS. P4$
                                  .IIF GE,<PRMS-3.>,PARGS. P3$
                                  .IIF GE,<PRMS-2.>,PARGS. P2$
                                  .IIF GE,<PRMS-1.>,PARGS. P1$
                                  .IF GE REGS$
                                    RSTR$ \REGS$
                                  .ENDC
                                  .RADIX 10
                                  .LIST
                                                                                CALL RERROR     ;ERROR # ERRN$'.
                                  .NLIST
                                  .RADIX 8
                                  .LIST
                                                                                .WORD ET$+ERRN
                                                                                .WORD <PRMS*10000>+MS$
                                  .NLIST
                                  ERRN=ERRN+1
                                .ENDM

                                .MACRO PARGS.,ADDR$
                                  .NTYPE PTYPE$,ADDR$
                                  .IF EQ,<PTYPE$&70>
                                    .IIF EQ,<PTYPE$&7>-REGS$,RSTR$ \REGS$
                                    .LIST
                                                                                MOV ADDR$,-(SP)
                                    .NLIST
                                  .IFF
                                    .IF EQ,<PTYPE$&7>-1         ;PICK A REGISTER TO USE
                                      REGU$=2                   ;SELECT R2 IF R1 IS USED IN PARAMETER FETCH
                                    .IFF
                                      REGU$=1                   ;OTHERWISE USE R1
                                    .ENDC
                                    .IF NE,<REGU$-REGS$>        ;IF REGISTER NOT ALREADY SAVED
                                      .IF GE,REGS$
                                        RSTR$ \REGS$            ;RESTORE CURRENT SAVED REGISTER
                                      .ENDC
                                      SAVR$ \REGU$              ;THEN SAVE SELECTED REGISTER
                                    .ENDC
                                    GETP$ \REGS$,ADDR$
                                  .ENDC
                                .ENDM
```

MACRO DEFINITIONS

```
      1                                     .MACRO SAVR$ REGN
      2                                       .LIST
      3                                                                   MOV R'REGN,SAVREG
      4                                       .NLIST
      5                                       REGS$=REGN
      6                                     .ENDM
      7
      8                                     .MACRO RSTR$ REGN
      9                                       .LIST
     10                                                                   MOV SAVREG,R'REGN
     11                                       .NLIST
     12                                       REGS$=-1
     13                                     .ENDM
     14
     15                                     .MACRO GETP$ REGN,ADDR$
     16                                       .LIST
     17                                                                   MOV ADDR$,R'REGN
     18                                                                   MOV R'REGN,-(SP)
     19                                       .NLIST
     20                                     .ENDM
```

```
                        ;
                        ;       PRIMARY ERROR REPORTING (TEST 4)
                        ;

                        .MACRO  SOFTER  NUM,ARGS
                        ERROR   ERSOFT,NUM,<ARGS>
                                                        MOV     #ERRMES,OUT.RQ
                        .ENDM

                        .MACRO  REPSFT  SFTFLG,ECCFG,SEKFLG
                        .IF     NB,SFTFLG
                                                        MOV     #1,OUT.02
                        .IFF
                                                        CLR     OUT.02
                        .ENDC
                        .IF     NB,ECCFG
                                                        MOV     #1,OUT.03
                        .IFF
                                                        CLR     OUT.03
                        .ENDC
                        .IF     NB,SEKFLG
                                                        MOV     #1,OUT.04
                        .IFF
                                                        CLR     OUT.04
                        .ENDC
                                                        PUSH    R0
                                                        MOV     U.UNUM(R5),R0
                                                        ADD     U.SUBU(R5),R0
                                                        MOV     R0,OUT.01
                                                        MOV     #T4SOFT,R0
                                                        CALL    HOSTRQ
                                                        POP     R0

                        .ENDM

                        .MACRO  HARDER  NUM,ARGS
                        ERROR   ERHARD,NUM,<ARGS>
                                                        MOV     #ERRMES,OUT.RQ
                        .ENDM

                        .MACRO  DEVFTL  NUM,ARGS
                        ERROR   FTLDEV,NUM,<ARGS>
                                                        MOV     #ERRMES,OUT.RQ
                        .ENDM

                        .MACRO  SYSFTL  NUM,ARGS
                        ERROR   FTLSYS,NUM,<ARGS>
                                                        MOV     #ERRMES,OUT.RQ
                        .ENDM

                        .MACRO  ERROR   TYPE,NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       5
                        MOVMSG  #MS'NUM,4
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
```

```
                                        .ENDC
                                                                MOV     LUNIT,OUT.03
                                                                MOV     #NUM!TYPE+3000.,R2
                                                                MOV     R2,OUT.02
                                                                MOV     #.,OUT.01

                                        .RADIX
                                        .ENDM

                                        .MACRO  CERROR  STNUM,ARGS
                                        .RADIX  10
                                NUMPTR  =       STNUM
                                        .IRP    X,<ARGS>
                                        MOVMSG  X,\NUMPTR
                                NUMPTR  =       NUMPTR + 1
                                        .ENDR
                                        .RADIX
                                        .ENDM

                                        .MACRO  ERRORC  ARGS
                                        .RADIX  10
                                        .IRP    X,<ARGS>
                                        MOVMSG  X,\NUMPTR
                                NUMPTR  =       NUMPTR + 1
                                        .ENDR
                                        .RADIX
                                        .ENDM

                                        .MACRO  MOVMSG  ARG,INDX

                                        .IF     LT,INDX-10
                                                                MOV     ARG,OUT.0'INDX
                                        .IFF
                                                                MOV     ARG,OUT.'INDX
                                        .ENDC
                                        .ENDM

                                        .MACRO  ENDERR  POS
                                        .RADIX  10
                                        .IF     NB,POS
                                        NDERR   POS
                                        .IFF
                                        NDERR   \NUMPTR
                                        .ENDC
                                        .RADIX
                                        .ENDM

                                        .MACRO  NDERR   POS
                                        .IF     NE,POS
                                        MOVMSG  #SER22,POS
                                                                MOV     #POS+1,ERRPOS   ; SET THE POSITION
                                        .IFF
                                                                CLR     ERRPOS          ; CLEAR THE POSITION
                                        .ENDC
                                        .ENDM
                                ;
                                ;       MESSAGE REPORTING MACRO
                                ;
```

```
                        .MACRO  MSSG    NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       3
                        MOVMSG  #MS'NUM,2
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC
                                                        PUSH    <R0,R1>
                                                        MOV     LUNIT,OUT.01
                                                        MOV     #MESSAG,R0
                                                        CALL    HOSTRQ
                                                        POP     <R1,R0>
                        .RADIX
                        .ENDM
                ;
                ;
                        .MACRO  MSSGE   NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       3
                        MOVMSG  #'NUM,2
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC
                                                        PUSH    <R0,R1>
                                                        MOV     #MESSAG,R0
                                                        CALL    HOSTRQ
                                                        POP     <R1,R0>
                        .RADIX
                        .ENDM
```

MACRO DEFINITIONS

```
      1                                 ;ASSUME MACRO
      2                                         .MACRO  ASSUME  P1,P2
      3                                         .IF     NE,P1-P2
      4                                         .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
      5                                         .ENDC
      6                                         .ENDM
```

```
                        ;RETURN DRIVE STATUS MACRO WITH ERROR REPORTING

                                .MACRO  DSTAT,LAB$,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   ME
                                .LIST
                                CALL    RDSTAT                  ; GET DRIVE STATUS
                                BIT     #10000,R1               ; SEE IF ANY ERRORS
                                BEQ     2$                      ; IF NO ERROR, BRANCH
                                BIT     #4000,R1                ; SEE IF XMIT ERROR
                                BEQ     1$                      ; IF SO, BRANCH
                                .NLIST  ME
                                .LIST   MEB
                                HARDER  E1                      ; REPORT INVALID STATUS ERROR
                                .NLIST  MEB
                                .LIST   ME
                                BR      LAB$                    ; BRANCH TO DONE
                        1$:
                                .NLIST  ME
                                .LIST   MEB
                                HARDER  E2                      ; REPORT XMIT ERROR
                                .NLIST  MEB
                                .LIST   ME
                                BR      LAB$                    ; BRANCH TO DONE
                        2$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

```
1                                       ;
2                                       ;       XOR THE CONTENTS OF TWO REGISTERS
3                                       ;
4                                               .MACRO  RXOR    REG1,REG2
5                                               MOV     REG2,-(SP)              ; SAVE REGISTER REG2
6                                               BIC     REG1,REG2               ; CLEAR COORESPONDING BITS IN REG2
7                                               BIC     (SP)+,REG1              ; CLEAR COORESPONDING BITS IN REG1
8                                               BIS     REG1,REG2               ; OR WHAT'S LEFT
9                                               .ENDM
```

```
                        ;
                        ;       CONVERT .BLKW CALLS TO ACTUAL WORDS GENERATED
                        ;
                        .MACRO  .BLKW   COUNT
                        .NLIST
                        .REPT   COUNT
                        .WORD   0
                        .ENDR
                        .LIST
                        .ENDM
```

MACRO DEFINITIONS

```
                        ;       SDI INTERCHANGE WITH DRIVE WITH ERROR REPORTING
                        .MACRO  TALKX   ERRLAB,E1,E2
                        .NLIST
                        .NLIST  MEB
                        .LIST   ME
                        .LIST
                        CALL    TALK                    ; INITIATE SDI INTERCHANGE
                        TST     R2                      ; SEE IF ERROR OCCURRED
                        BEQ     12$                     ; IF NOT, BRANCH
                        BPL     11$                     ; IF SO, BRANCH
                        .NLIST  ME
                        .LIST   MEB
                        HARDER  E1                      ;SEND COMMAND ERROR
                        .NLIST  MEB
                        .LIST   ME
                        BR      ERRLAB
                11$:
                        .NLIST  ME
                        .LIST   MEB
                        HARDER  E2                      ;RECEIVE COMMAND ERROR
                        .NLIST  MEB
                        .LIST   ME
                        BR      ERRLAB
                12$:
                        .NLIST
                        .NLIST  ME
                        .LIST   MEB
                        .LIST
                        .ENDM
```

```
                        .SBTTL  MACRO FOR OVERLAY TABLE
                        .MACRO  DFOVLY  LABL$,ONAME,SAREA,EAREA
                        .WORD   0
                        .WORD   OVL.'ONAME'*4
                        .IF     NE,OCNT$-LABL$
                        .ERROR  ; OVERLAY NUMBER AND POSITION IN TABLE DO NOT MATCH
                        .ENDC
OCNT$   =       OCNT$+1
                        .ENDM

                        .MACRO  MESSAGES
                        .SBTTL  MESSAGES
;MESSAGE STORAGE OVERLAY


                        DMOVLY MS,0
                        .NLIST BEX

MS1:    .ASCII\''TIME-OUT ON SEND'N\
                        .ASCII\R1R1\
                        .BYTE 0
MS2:    .ASCII\''TIME-OUT ON RECEIVE'N\
                        .ASCII\R1R1\
                        .BYTE 0
MS3:    .ASCII\''FIRST WORD RECEIVED WAS NOT A START FRAME'N\
                        .ASCII\R1R1\
                        .BYTE 0
MS4:    .ASCII\''FRAMING ERROR ON LEVEL 0 RESPONSE'N\
                        .ASCII\R1R1\
                        .BYTE 0
MS5:    .ASCII\''CHECKSUM ERROR ON LEVEL 0 RESPONSE'N\
                        .ASCII\R1R1\
                        .BYTE 0
MS6:    .ASCII\''RESPONSE LONGER THAN EXPECTED'N\
                        .ASCII\R1R1\
                        .BYTE 0
MS7:    .ASCII\''CODE FROM RECEIVE WAS UNINTELLIGIBLE FROM SUBSYSTEM = 'H16N\
                        .ASCII\R1R1\
                        .BYTE 0
MS8:    .ASCII\''COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
                        .ASCII\R1\
                        .ASCII\''  EXPECTED RESPONSE  'H8N\
                        .ASCII\''  ACTUAL RESPONSE    'H8N\
                        .ASCII\R1\
                        .BYTE 0
MS9:    .ASCII\''DRIVE NOT ASSERTING RECEIVER READY IN DRIVE STATE'N\
                        .BYTE 0
MS10:   .ASCII\''FAILED TO RECEIVE VALID DRIVE STATE'NR1\
                        .BYTE 0
MS11:   .ASCII\''CANNOT RECEIVE DRIVE STATE FROM DRIVE'N\
                        .ASCII\''CHECK IF DRIVE IS POWERED ON.'NR1\
                        .BYTE 0
MS12:   .ASCII\''DRIVE STATE RECEIVED HAS BAD PARITY'NR1\
                        .BYTE 0
MS13:   .ASCII\''NO VALID STATE FROM DRIVE'NR1\
                        .BYTE 0
MS14:   .ASCII  \''SUBUNIT CHARACTERISTICS SAY THERE ARE ZERO READ ONLY GROUPS'N\
```

```
                                        .ASCII  \'IN THE DIAGNOSTIC AREA'N\
                                        .BYTE   0
                                MS15:   .ASCII  \'SUBUNIT CHARACTERISTICS SAY THERE ARE LESS THAN 1 READ/WRITE'N\
                                        .ASCII  \'GROUPS IN THE DIAGNOSTIC AREA'N\
                                        .BYTE   0
                                MS16:   .ASCII\'NEITHER R/W READY NOR ATTENTION SET AFTER RECALIBRATE COMMAND'NR1\
                                        .BYTE 0
                                MS17:   .ASCII\'SUBUNIT CHARACTERISTICS SAY LESS THAN 1 DIAGNOSTIC CYLINDER'N\
                                        .BYTE 0
                                MS18:   .ASCII\'READ/WRITE READY DROPPED BEFORE FORMAT OPERATION'N\
                                        .BYTE 0
                                MS19:   .ASCII\'FORMAT OPERATION REPORTED TIME-OUT FAILURE'N\
                                        .ASCII\'  CYLINDER 'D28'.   GROUP  'D8'.  TRACK 'D8'.'N\
                                        .BYTE 0
                                MS20:   .ASCII\'AFTER RECAL, ERROR BITS WERE SET'NR1\
                                        .BYTE 0
                                MS21:   .ASCII\N'LOGGABLE INFORMATION AFTER RECAL'NR1\
                                        .BYTE 0
                                MS22:   .ASCII\'READ/WRITE READY DROPPED BEFORE WRITE OPERATION'N\
                                        .BYTE 0
                                MS23:   .ASCII\'COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
                                        .ASCII\'WRITE OPERATION REPORTED FAILURE -- ERROR CODE 'O8' OCTAL.'N\
                                        .ASCII\'DBN 'D24'.  CYLINDER 'D28'.  GROUP 'D8'.  TRACK 'D8'.'N\
                                        .BYTE 0
                                MS24:   .ASCII\'READ/WRITE READY DROPPED BEFORE READ OPERATION'N\
                                        .BYTE 0
                                MS25:   .ASCII\'COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
                                        .ASCII\'READ OPERATION REPORTED FAILURE -- ERROR CODE 'O8' OCTAL.'N\
                                        .ASCII\'DBN 'D24'.  CYLINDER 'D28'.  GROUP 'D8'.  TRACK 'D8'.'N\
                                        .BYTE 0
                                MS26:   .ASCII\'COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
                                        .ASCII\'DATA COMPARE FAILURE ON WORD 'D16'.'N\
                                        .ASCII\'EXPECTED DATA  'H16N\
                                        .ASCII\'ACTUAL DATA    'H16N\
                                        .ASCII\'CYLINDER 'D28'.  GROUP 'D8'.  TRACK 'D8'.'N\
                                        .BYTE 0
                                MS27:   .ASCII\'SEEK COMPLETE TIME-OUT -- READ/WRITE READY DID NOT SET.'N\
                                        .ASCII\'SEEK WAS TO CYLINDER 'D28'.  GROUP 'D8'.'N\
                                        .BYTE 0
                                MS28:   .ASCII\'NO BLOCK ON THIS TRACK CAN BE READ. LAST BLOCK TRIED:'N\
                                        .ASCII\A1'BN 'D24'.  CYLINDER 'D28'.  GROUP 'D8'.  TRACK 'D8'.'N\
                                        .BYTE 0
                                MS29:   .ASCII  \'AVAILABLE WAS NOT ASSERTED AFTER DISCONNECT'N\
                                        .ASCII  \'  STATE RECEIVED  'H16N\
                                        .BYTE   0
                                MS30:   .ASCII\'INVALID COMMAND 'H16' WAS SUCCESSFUL'N\
                                        .BYTE   0
                                MS31:   .ASCII\'COMMAND WITH 'R1' LENGTH = 'D8' WAS SUCCESSFUL'N\
                                        .BYTE   0
                                MS32:   .ASCII\'UNIT DID NOT REPORT TRANSMITTION ERROR'NR1\
                                        .BYTE   0
                                MS33:   .ASCII\'UNIT ACCEPTED AN INVALID GROUP NUMBER FROM GROUP SELECT LEVEL 1'N\
                                        .BYTE   0
                                MS34:   .ASCII\'UNABLE TO CORRECTLY READ OVERLAY 'O3NR1\
                                        .BYTE   0
                                MS35:   .ASCII\'SUCCESSFULLY WROTE IN DBN AREA WHEN DRIVE WAS WRITE PROTECTED'N\
                                        .BYTE   0
```

```
                                MS36:   .ASCII  \'DRIVE IS NOT PROPERLY FORMATTED.'NR1\
                                        .BYTE   0
                                MS37:   .ASCII  \'DRIVE IS FORMATTED IN 576 BYTE MODE.'N\
                                        .ASCII  \'TO RUN WITH A UDA, THIS DRIVE NEEDS TO BE FORMATTED '\
                                        .ASCII  \'IN 512 BYTE MODE.'N\
                                        .BYTE   0
                                MS38:   .ASCII  \'NO COPY OF THE FCT COULD BE READ.'NR1\
                                        .BYTE   0
                                SER36:  .ASCII  \'UDA WILL SPIN DOWN THIS DRIVE IF USED IN NORMAL SYSTEM OPERATION.'N\
                                        .ASCII  \'THIS DRIVE NEEDS TO BE FORMATTED.'\
                                        .BYTE   0
                                SER39:  .ASCII\'THIS UDA AND ALL DRIVES ATTACHED WILL BE REMOVED FROM TESTING'N\
                                        .BYTE   0
                                SER00:  .ASCII\'COMMAND WAS 'R1\
                                        .BYTE   0
                                MS.ONL: .ASCII\'ONLINE'N\
                                        .BYTE   0
                                MS.CLR: .ASCII\'DRIVE CLEAR'N\
                                        .BYTE   0
                                MS.DIS: .ASCII\'DISCONNECT'N\
                                        .BYTE   0
                                MS.GCR: .ASCII\'GET COMMON CHARACTERISTICS'N\
                                        .BYTE   0
                                MS.SCR: .ASCII\'GET SUBUNIT CHARACTERISTICS'N\
                                        .BYTE   0
                                MS.GST: .ASCII\'GET STATUS'N\
                                        .BYTE   0
                                MS.MOD: .ASCII\'CHANGE MODE'N\
                                        .BYTE   0
                                MS.SEK: .ASCII\'SEEK'N\
                                        .BYTE   0
                                MS.INR: .ASCII\'INITIATE RECALIBRATE'N\
                                        .BYTE   0
                                MS.RUN: .ASCII\'SPIN UP'N\
                                        .BYTE   0
                                MS2000: .ASCII  \'UNABLE FIND REQUESTED DRIVE FOR TESTING'N\
                                        .ASCII  \'THE FOLLOWING IS VISIBLE ON THE PORTS'N\
                                        .ASCII  \'UDA PORT 0 -- 'R1\
                                        .ASCII  \'UDA PORT 1 -- 'R1\
                                        .ASCII  \'UDA PORT 2 -- 'R1\
                                        .ASCII  \'UDA PORT 3 -- 'R1\
                                        .BYTE   0
                                SER10:  .ASCII  \'NO DRIVE ATTACHED'N\
                                        .BYTE   0
                                SER11:  .ASCII  \'RCVR RDY NEVER ASSERTED'N\
                                        .BYTE   0
                                SER12:  .ASCII  \'TIMEOUT OF SEND'N\
                                        .BYTE   0
                                SER13:  .ASCII  \'TIMEOUT OF RECEIVE'N\
                                        .BYTE   0
                                SER14:  .ASCII  \'FIRST WORD RECEIVED WAS NOT START FRAME'N\
                                        .BYTE   0
                                SER15:  .ASCII  \'FRAMING ERROR ON LEVEL 0 RECEIVE'N\
                                        .BYTE   0
                                SER16:  .ASCII  \'CHECKSUM ERROR ON LEVEL 0 RECEIVE'N\
                                        .BYTE   0
                                SER17:  .ASCII  \'RESPONSE LONGER THAN EXPECTED FOR GET STATUS CMD'N\
```

```
                                .BYTE   0
                        SER18:  .ASCII  \''DRIVE ''R1\
                                .BYTE   0
                        SER18D: .ASCII  \D6'', ''\
                        SER18C: .ASCII  \D6'', ''\
                        SER18B: .ASCII  \D6'', ''\
                        SER18A: .ASCII  \D6N\
                                .BYTE   0
                        SER22:  .ASCII\''REAL TIME STATE ''H16N\
                                .ASCII\''STATLS (R TO L): ''H16S2H16S2H16S2H16S2H16S2H16S2H16N\
                                .BYTE   0
                        SER23:  .ASCII\''REAL TIME STATE ''H16N\
                                .BYTE   0
                        SER40:  .ASCII  \''DRIVE NOT AVAILABLE TO THIS UDA'N\
                                .BYTE   0
                        SER41:  .ASCII  \''DRIVE NOT SPINABLE'N\
                                .BYTE   0
                        SER50:  .ASCII\''COMMAND''\
                                .BYTE   0
                        SER51:  .ASCII\''RESPONSE''\
                                .BYTE   0
                        SER52:  .ASCII\''WHEN A CONTINUE OR END FRAME DID NOT FOLLOW A START FRAME'N\
                                .BYTE   0
                        SER53:  .ASCII\''WHEN AN END FRAME WAS SENT WITH NO START FRAME'N\
                                .BYTE   0
                        SER54:  .ASCII\''WHEN AN END FRAME WITH A BAD CHECKSUM WAS SENT'N\
                                .BYTE   0
                        SER55:  .ASCII\''WHEN A CONTINUE FRAME WAS SENT WITH NO START FRAME'N\
                                .BYTE   0
                        SER56:  .ASCII\''WHEN TWO CONSECUTIVE START FRAMES WERE SENT'N\
                                .BYTE   0
                        SER57:  .ASCII\''WHEN AN END FRAME WAS SENT AFTER A START FRAME TIMED OUT'N\
                                .BYTE   0

                        OVL.MS  = .
                                .ENDM
```

```
                                        .SBTTL  RTDS   - REAL TIME DRIVE STATE ROUTINE WITH ERROR REPORTING (TEST 4)
                                ;
                                ;       DIFF BETWEEN SEND + RECEIVE ERRORS, REPORT HERE
                                ;
000720  020746                 RTDS:   CALL    RTDSL           ; GET REAL TIME DRIVE STATE
000721  115002                          TST     R2              ; SEE IF ERROR OCCURRED
000722  010745                          BEQ     1$              ; IF NOT, BRANCH
000723                                  DEVFTL  13              ; REPORT DEVICE FATAL ERROR
000723  104200  000461  001037                                          MOV     #MS13,OUT.04
000726  104300  001475  001036                                          MOV     LUNIT,OUT.03
000731  104202  045705                                                  MOV     #13!FTLDEV+3000.,R2
000733  104020  001035                                                  MOV     R2,OUT.02
000735  104200  000735  001034                                          MOV     #.,OUT.01
000740  104200  060013  001033                                          MOV     #ERRMES,OUT.RQ
000743                                  ENDERR  0
000743  114000  003116                                                  CLR     ERRPOS          ; CLEAR THE POSITION
000745  000000                  1$:     RETURN

                                        .SBTTL  RTDSL - CALL RDSTAT
000746                          RTDSL:
000746  104302  002646                  MOV     SDI,R2                  ; GET INTERCONNECT CODE
000750  020754                          CALL    RDSTAT
000751  103201  077674                  BIC     #077674,R1              ; CLEAR UNUSED BITS
000753  000000                          RETURN

                                        .SBTTL  RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
000754                          RDSTAT:
                                ;
                                ;RETURN DRIVE STATUS
                                ;STATUS RETURNED IN DM REGISTER 1
                                ;
                                ;       INPUT R2 MUST HAVE INTERCONNECT CODE
                                ;
000754                                  PUSH    <R3,R0>                 ; SAVE R3 AND R0
000754  100463                                                                  MOV R3,-(SP)
000755  100467                                                                  MOV R0,-(SP)
000756  104203  024000                  MOV     #24000,R3               ; ERROR COUNT
000760  060007          1$:     XFC     STATUS                  ; GET DRIVE'S STATUS
000761  102201  000004                  BIT     #DCLOCK,R1              ; SE IF DRIVE CLOCK IS PRESENT
000763  010775                          BEQ     3$              ; IF NOT, BRANCH
000764  102201  000400                  BIT     #RCVERR,R1              ; RECIEVER ERRORS
000766  010774                          BEQ     2$              ; IF NOT VALID, BRANCH
000767  117403                          DEC     R3              ; DECREMENT ERROR COUNT
000770  050760                          BNE     1$              ; IF ERROR COUNT NON-ZERO, BRANCH
000771  104202  177777                  MOV     #177777,R2              ; MARK AS RECEIVE ERROR
000773  000775                          BR      3$
000774  114002          2$:     CLR     R2                      ; NO ERRORS
000775                  3$:     POP     <R0,R3>                 ; RESTORE R0, R3
000775  104267                                                                  MOV (SP)+,R0
000776  104263                                                                  MOV (SP)+,R3
000777  000000                          RETURN                  ; RETURN TO CALLING MODULE
```

```
                                        .SBTTL  HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
001000                          HOSTRQ:
                                ;
                                ;SEND REQUEST BUFFER TO HOST AND WAIT FOR RESPONSE.
                                ;CLEAR ARGUMENT AREA OF OUT BUFFER IN PREPARATION
                                ;FOR NEXT HOSTRQ CALL.
                                ;
                                ;INPUTS:
                                ;       R0 - HOST REQUEST NUMBER
                                ;       OUT BUFFER LOADED WITH DATA

001000                          PUSH    <R0,R1,R2>
001000  100467                                                          MOV R0,-(SP)
001001  100461                                                          MOV R1,-(SP)
001002  100462                                                          MOV R2,-(SP)
001003  104300  001475  001036        MOV     LUNIT,OUT.03
001006  104070  001033              MOV     R0,OUT.RQ               ; STORE REQUEST NUMBER IN BUFFER
001010  104207  001033      SNDAGN: MOV     #OUT.RQ,R0              ; SEND BUFFER TO HOST
001012  104201  000043              MOV     #BUFSIZ,R1
001014  060016                      XFC     MRD
001015  115001                      TST     R1                      ; CHECK FOR ERROR
001016  051010                      BNE     SNDAGN                  ; IF ERROR, TRY AGAIN
001017  104207  001033              MOV     #OUT.RQ,R0              ; WAIT FOR RESPONSE FROM HOST
001021  104201  000043              MOV     #BUFSIZ,R1
001023  060017                      XFC     MWR
001024  104200  177777  001033      MOV     #-1,OUT.RQ              ; MAKE REQUEST ILLEGAL
001027                          POP     <R2,R1,R0>
001027  104262                                                          MOV (SP)+,R2
001030  104261                                                          MOV (SP)+,R1
001031  104267                                                          MOV (SP)+,R0
001032  000000                          RETURN
```

```
                                ;STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS
                                ;
                                ;OUT BUFFER - DATA TO SEND TO HOST
                                ;
001033  000000  OUT.RQ: .WORD 0          ;HOST REQUEST CODE
001034  000000  OUT.01: .WORD 0          ;DATA ARGUMENT  1
001035  000000  OUT.02: .WORD 0          ;DATA ARGUMENT  2
001036  000000  OUT.03: .WORD 0          ;DATA ARGUMENT  3
001037  000000  OUT.04: .WORD 0          ;DATA ARGUMENT  4
001040  000000  OUT.05: .WORD 0          ;DATA ARGUMENT  5
001041  000000  OUT.06: .WORD 0          ;DATA ARGUMENT  6
001042  000000  OUT.07: .WORD 0          ;DATA ARGUMENT  7
001043  000000  OUT.08: .WORD 0          ;DATA ARGUMENT  8
001044  000000  OUT.09: .WORD 0          ;DATA ARGUMENT  9
001045  000000  OUT.10: .WORD 0          ;DATA ARGUMENT 10
001046  000000  OUT.11: .WORD 0          ;DATA ARGUMENT 11
001047  000000  OUT.12: .WORD 0          ;DATA ARGUMENT 12
001050  000000  OUT.13: .WORD 0          ;DATA ARGUMENT 13
001051  000000  OUT.14: .WORD 0          ;DATA ARGUMENT 14
001052  000000  OUT.15: .WORD 0          ;DATA ARGUMENT 15
001053  000000  OUT.16: .WORD 0          ;DATA ARGUMENT 16
001054  000000  OUT.17: .WORD 0          ;DATA ARGUMENT 17
001055  000000  OUT.18: .WORD 0          ;DATA ARGUMENT 18
001056  000000  OUT.19: .WORD 0          ;DATA ARGUMENT 19
001057  000000  OUT.20: .WORD 0          ;DATA ARGUMENT 20
001060  000000  OUT.21: .WORD 0          ;DATA ARGUMENT 21
001061  000000  OUT.22: .WORD 0          ;DATA ARGUMENT 22
001062  000000  OUT.23: .WORD 0          ;DATA ARGUMENT 23
001063  000000  OUT.24: .WORD 0          ;DATA ARGUMENT 24
001064  000000  OUT.25: .WORD 0          ;DATA ARGUMENT 25
001065  000000  OUT.26: .WORD 0          ;DATA ARGUMENT 26
001066  000000  OUT.27: .WORD 0          ;DATA ARGUMENT 27
001067  000000  OUT.28: .WORD 0          ;DATA ARGUMENT 28
001070  000000  OUT.29: .WORD 0          ;DATA ARGUMENT 29
001071  000000  OUT.30: .WORD 0          ;DATA ARGUMENT 30
001072  000000  OUT.31: .WORD 0          ;DATA ARGUMENT 31
001073  000000  OUT.32: .WORD 0          ;DATA ARGUMENT 32
001074  000000  OUT.33: .WORD 0          ;DATA ARGUMENT 33
001075  000000  OUT.34: .WORD 0          ;DATA ARGUMENT 34
        000043  BUFSIZ  =       . - OUT.RQ  ;SIZE OF BUFFER
```

```
                                        .SBTTL  TALK   - SDI LEVEL 2 INTERCHANGE ROUTINE
001076                          TALK:
                                ;
                                ;       TALK SENDS THE COMMAND TO THE DRIVE.  IF AN ERROR OCCURRS, R2 IS
                                ;       RETURNED NONZERO
                                ;
001076                                  PUSH    <R4,R5>         ; SAVE POINTER TO UNIT AND SUBUNIT PARAMETERS
001076  100464                                                                  MOV R4,-(SP)
001077  100465                                                                  MOV R5,-(SP)
001100  104302  002646                  MOV     SDI,R2          ; GET UNIT SDI SELECT MASK
001102  104205  000001                  MOV     #1,R5           ; SEND TIMEOUT DEFAULT IS ONE
001104  104207  002666                  MOV     #SNDONE,R0      ; MOVE SNDONE ADDRESS TO R0
001106  106037                          CMP     R3 R0           ; SEE IF ONLY TO BE SENT ONCE
001107  041112                          BCC     10$             ; IF SO, BRANCH
001110  104205  001750                  MOV     #MAXSND,R5      ; SEND MAXIMUM NUMBER OF TIMES (ONLINE)
001112  104137                  10$:    MOV     (R3),R0         ; POINTS TO SDI COMMAND BUFFER
001113  104631  000001                  MOV     1(R3),R1        ; LOAD BYTE COUNT
001115  060004                          XFC     SEND            ; SEND SDI COMMAND
001116  115001                          TST     R1              ; SEE IF SDI COMMAND SENT SUCESSFULLY
001117  011145                          BEQ     15$             ; IF SO, BRANCH
001120  117405                          DEC     R5              ; DECREMENT TIMEOUT
001121  051112                          BNE     10$             ; IF UNEXPIRED, BRANCH
001122                                  POP     R5              ; RESTORE R5
001122  104265                                                                  MOV (SP)+,R5
001123                                  PUSH    R3              ; SAVE SDI PACKET POINTER
001123  100463                                                                  MOV R3,-(SP)
001124                                  HARDER  1
001124  104200  000000  001037                                         MOV     #MS1,OUT.04
001127  104300  001475  001036                                         MOV     LUNIT,OUT.03
001132  104202  105671                                                 MOV     #1!ERHARD+3000.,R2
001134  104020  001035                                                 MOV     R2,OUT.02
001136  104200  001136  001034                                         MOV     #.,OUT.01
001141  104200  060013  001033                                         MOV     #ERRMES,OUT.RQ
001144  001335                          BR      50$             ; BRANCH
001145                          15$:    POP     R5              ; RESTORE R5
001145  104265                                                                  MOV (SP)+,R5
001146  106203  002717                  CMP     #LONG,R3        ; SEE IF LONG TIMEOUT TO BE USED
001150  071154                          BMI     20$             ; IF SO, BRANCH
001151  104304  001461                  MOV     SDISTO,R4       ; R4 HAS SHORT TIMEOUT
001153  001156                          BR      25$             ; BRANCH
001154  104304  001462          20$:    MOV     SDILTO,R4       ; R4 HAS LONG TIMEOUT
001156  104637  000002          25$:    MOV     2(R3),R0        ; POINT TO RECEIVE BUFFER
001160  104631  000003                  MOV     3(R3),R1        ; NUMBER OF WORDS IN RESPONSE
001162                                  PUSH    R3              ; SAVE POINTER TO COMMAND
001162  100463                                                                  MOV R3,-(SP)
001163  060005                          XFC     RCV             ; RECEIVE SDI RESPONSE
001164                                  POP     R3              ; RESTORE R3
001164  104263                                                                  MOV (SP)+,R3
001165  115001                          TST     R1              ; SEE IF SDI RESPONSE RECEIVED SUCESSFULLY
001166  011412                          BEQ     60$             ; IF SO, BRANCH
001167  106201  000001                  CMP     #1,R1           ; SEE IF TIMEOUT
001171  051216                          BNE     30$             ; IF NOT, BRANCH
001172  117404                          DEC     R4              ; DECREMENT TIMEOUT VALUE
001173  051156                          BNE     25$             ; IF TIMEOUT UNEXPIRED, BRANCH
001174                                  PUSH    R3              ; SAVE R3
001174  100463                                                                  MOV R3,-(SP)
001175                                  HARDER  2
```

```
        001175  104200  000014  001037                          MOV     #MS2,OUT.04
        001200  104300  001475  001036                          MOV     LUNIT,OUT.03
        001203  104202  105672                                  MOV     #2!ERHARD+3000.,R2
        001205  104020  001035                                  MOV     R2,OUT.02
        001207  104200  001207  001034                          MOV     #.,OUT.01
        001212  104200  060013  001033                          MOV     #ERRMES,OUT.RQ
    44  001215  001335                      BR      50$         ; BRANCH
    45  001216  106201  000002      30$:    CMP     #2,R1       ; SEE IF FIRST WORD NOT START FRAME
    46  001220  051242                      BNE     35$         ; IF NOT, BRANCH
    47  001221                              HARDER  3
        001221  104200  000032  001037                          MOV     #MS3,OUT.04
        001224  104300  001475  001036                          MOV     LUNIT,OUT.03
        001227  104202  105673                                  MOV     #3!ERHARD+3000.,R2
        001231  104020  001035                                  MOV     R2,OUT.02
        001233  104200  001233  001034                          MOV     #.,OUT.01
        001236  104200  060013  001033                          MOV     #ERRMES,OUT.RQ
    48  001241  001335                      BR      50$         ; BRANCH
    49  001242  106201  000004      35$:    CMP     #4,R1       ; SEE IF FRAMING ERROR
    50  001244  051266                      BNE     40$         ; IF NOT, BRANCH
    51  001245                              HARDER  4
        001245  104200  000063  001037                          MOV     #MS4,OUT.04
        001250  104300  001475  001036                          MOV     LUNIT,OUT.03
        001253  104202  105674                                  MOV     #4!ERHARD+3000.,R2
        001255  104020  001035                                  MOV     R2,OUT.02
        001257  104200  001257  001034                          MOV     #.,OUT.01
        001262  104200  060013  001033                          MOV     #ERRMES,OUT.RQ
    52  001265  001335                      BR      50$         ; BRANCH
    53  001266  106201  000010      40$:    CMP     #10,R1      ; SEE IF CHECKSUM ERROR
    54  001270  051312                      BNE     45$         ; IF NOT, BRANCH
    55  001271                              HARDER  5
        001271  104200  000110  001037                          MOV     #MS5,OUT.04
        001274  104300  001475  001036                          MOV     LUNIT,OUT.03
        001277  104202  105675                                  MOV     #5!ERHARD+3000.,R2
        001301  104020  001035                                  MOV     R2,OUT.02
        001303  104200  001303  001034                          MOV     #.,OUT.01
        001306  104200  060013  001033                          MOV     #ERRMES,OUT.RQ
    56  001311  001335                      BR      50$         ; BRANCH
    57  001312  106201  000020      45$:    CMP     #20,R1      ; SEE IF BUFFER TOO SMALL
    58  001314  051353                      BNE     55$         ; IF NOT, BRANCH
    59  001315                              HARDER  6
        001315  104200  000135  001037                          MOV     #MS6,OUT.04
        001320  104300  001475  001036                          MOV     LUNIT,OUT.03
        001323  104202  105676                                  MOV     #6!ERHARD+3000.,R2
        001325  104020  001035                                  MOV     R2,OUT.02
        001327  104200  001327  001034                          MOV     #.,OUT.01
        001332  104200  060013  001033                          MOV     #ERRMES,OUT.RQ
    60  001335                      50$:    CERROR  5,<#SER00,COMND>
        001335  104200  002573  001040                          MOV     #SER00,OUT.05
        001340  104300  002647  001041                          MOV     COMND,OUT.06
    61  001343                              ENDERR  7           ; FLAG END OF REPORTING BUFFER
        001343  104200  003276  001042                          MOV     #SER22,OUT.07
        001346  104200  000010  003116                          MOV     #7+1,ERRPOS     ; SET THE POSITION
    62  001351                              POP     R3          ; SAVE SDI PACKET POINTER
        001351  104263                                                  MOV (SP)+,R3
    63  001352  001457                      BR      65$         ; EXIT
    64  001353                      55$:    HARDER  7           ; UNKNOWN ERROR RETURNED BY UDA
        001353  104200  000160  001037                          MOV     #MS7,OUT.04
```

```
        001356  104300  001475  001036                                  MOV     LUNIT,OUT.03
        001361  104202  105677                                          MOV     #7!ERHARD+3000.,R2
        001363  104020  001035                                          MOV     R2,OUT.02
        001365  104200  001365  001034                                  MOV     #.,OUT.01
        001370  104200  060013  001033                                  MOV     #ERRMES,OUT.RQ
    65  001373                          CERROR  5,<R1,#SER00,COMND>     ; REPORT ERROR
        001373  104010  001040                                          MOV     R1,OUT.05
        001375  104200  002573  001041                                  MOV     #SER00,OUT.06
        001400  104300  002647  001042                                  MOV     COMND,OUT.07
    66  001403                          ENDERR  8               ; FLAG END OF REPORTING BUFFER
        001403  104200  003276  001043                                  MOV     #SER22,OUT.08
        001406  104200  000011  003116                                  MOV     #8+1,ERRPOS     ; SET THE POSITION
    67  001411  001457                  BR      65$             ; EXIT
    68  001412  114002          60$:    CLR     R2              ; FLAG AS NO ERROR OCCURED
    69  001413  106637  000004          CMP     4(R3),R0        ; SEE IF COMMAND ACCEPTED
    70  001415  011457                  BEQ     65$             ; IF SO, BRANCH
    71  001416                          HARDER  8
        001416  104200  000221  001037                                  MOV     #MS8,OUT.04
        001421  104300  001475  001036                                  MOV     LUNIT,OUT.03
        001424  104202  105700                                          MOV     #8!ERHARD+3000.,R2
        001426  104020  001035                                          MOV     R2,OUT.02
        001430  104200  001430  001034                                  MOV     #.,OUT.01
        001433  104200  060013  001033                                  MOV     #ERRMES,OUT.RQ
    72  001436                          CERROR  5,<#SER00,COMND,4(R3),R0> ; REPORT FURTHER ERRORS
        001436  104200  002573  001040                                  MOV     #SER00,OUT.05
        001441  104300  002647  001041                                  MOV     COMND,OUT.06
        001444  104630  000004  001042                                  MOV     4(R3),OUT.07
        001447  104070  001043                                          MOV     R0,OUT.08
    73  001451                          ENDERR  9
        001451  104200  003276  001044                                  MOV     #SER22,OUT.09
        001454  104200  000012  003116                                  MOV     #9+1,ERRPOS     ; SET THE POSITION
    74  001457          65$:    POP     R4              ; RESTORE R4
        001457  104264                                                          MOV (SP)+,R4
    75  001460  000000                  RETURN
    76
    77  001461  000012          SDISTO: .WORD   10.             ; SDI SHORT TIMEOUT
    78  001462  000024          SDILTO: .WORD   20.             ; SDI LONG TIMEOUT
```

```
                                                .SBTTL  TO - CALCULATE TIMEOUT INTERVALS
001463                                  TO:
                                        ;
                                        ;       CALCULATE THE TIMEOUT IN 9SEC INTERVALS (SDI RECEIVE XFC TAKES
                                        ;       9 SEC)
                                        ;
001463  104201  000001                          MOV     #1,R1           ; SET UP LOG2 SHIFTER
001465  105011                         1$:     ADD     R1,R1           ; DOUBLE THE TIMEOUT VALUE
001466  117407                                 DEC     R0              ; DECREMENT COUNT
001467  051465                                 BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
001470  115407                         2$:     INC     R0              ; INCREMENT 9 SEC COUNT
001471  107201  000011                         SUB     #9.,R1          ; SUBTRACT 9 SEC FROM TIMEOUT
001473  031470                                 BPL     2$              ; IF MORE TIME TO GO, BRANCH
001474  000000                                 RETURN                  ; RETURN TO CALLING PROGRAM

001475  177777                  LUNIT:  .WORD -1                        ;LOGICAL UNIT NUMBER (-1 FOR NOT AVAILABLE)

                                                .SBTTL  LINIT - INIT THE DRIVE
001476  104302  002646          LINIT:  MOV     SDI,R2          ; R2 HAS INTERCONNECT CODE
001500  060011                         XFC     DINIT
001501  000000                         RETURN
```

```
                                        .SBTTL  STACK AREA
                                ;STACK AREA
                                
001502  123456                          .WORD 123456                    ;END MARKER FOR STACK
001503                                  .BLKW 31.                       ;STACK
001542  123456                  STACK:  .WORD 123456                    ;MARKER FOR STACK UNDERFLOW
                                
        003006                  SUB     =       SUB + CR                ;MODIFY SUB TO POINT AT SUBUNIT CHAR
        005670                          ERRN=3000.                      ;START ERROR NUMBERS AT 3000.

001543  003274                  SER18E: .WORD   SER18A                  ; FOR MULTIPLE SUBUNIT ERROR REPORTING
001544  003271                          .WORD   SER18B
001545  003266                          .WORD   SER18C
001546  003263                          .WORD   SER18D

                                ;
                                ;       EACH OVERLAY TABLE IS COMPOSED OF TWO WORDS:
                                ;
                                ;       0)      LOW ORDER STARTING ADDRESS OF OVERLAY
                                ;       1)      <1::0> HI ORDER STARTING UNIBUS ADDRESS
                                ;               <15::2> LENGTH OF OVERLAY
                                ;
        000000                  MSSG$   =       0
        000000                  OCNT$   =       0
001547                          OTABLE: DFOVLY  MSSG$,MS
001547  000000                          .WORD   0
001550  017354                          .WORD   OVL.MS*4
001551                                  DFOVLY  SETUP,SU
001551  000000                          .WORD   0
001552  005070                          .WORD   OVL.SU*4
001553                                  DFOVLY  TEST,TS
001553  000000                          .WORD   0
001554  011460                          .WORD   OVL.TS*4
        000003                  NUMOVL  =       <. - OTABLE> / 2
```

```
                                        .SBTTL  TEST 3 START AND LOOP ON UNITS
                                ;SEQUENCE THE DIAGNOSTICS TO ALL UNITS SELECTED.
                                ;TEST CODE WILL BE CALLED FOR EACH DISK SUBUNIT TO BE TESTED.
                                ;  LUNIT WILL CONTAIN LOGICAL UNIT NUMBER OF DRIVE FOR ERROR REPORTS
                                ;  SDI WILL CONTAIN SDI INTERCONNECT CODE FOR SELECTED DRIVE
                                ;  UNITNB WILL CONTAIN AN EVEN NUMBER FOR TESTING FIRST SUBUNIT OF A DRIVE
                                ;                      AN ODD NUMBER FOR TESTING SECOND SUBUNIT OF A DRIVE
                                
001555  114001          T3STRT: CLR     R1              ;START WITH UNIT 0 INDEX
                                
001556  104207  002625  PORT2:  MOV     #UNITS,R0       ; GET POINTER TO UNITS TABLE
001560  105017                  ADD     R1,R0           ; ADD INDEX
001561  104173                  MOV     (R0),R3         ; GET CONTENTS OF TABLE
001562  031571                  BPL     1$              ; IF THIS UNIT IS PRESENT, BRANCH
001563  102201  000003          BIT     #3,R1           ; SEE IF ON SUBUNIT 0 OF UNIT
001565  051623                  BNE     PORT5           ; IF NOT, TEST NEXT SUBUNIT
001566  105201  000003          ADD     #3,R1           ; IF NO SUBUNIT 0, THEN NO UNIT - SKIP OTHER SUBUNITS
001570  001623                  BR      PORT5           ; BYPASS IF NO UNIT
001571  102203  040000  1$:     BIT     #40000,R3       ; SEE IF THIS UNIT IS TO BE TESTED
001573  051623                  BNE     PORT5           ; IF NOT, BRANCH
001574  104010  002645          MOV     R1,UNITNB       ; STORE UNIT INDEX
001576  104030  001475          MOV     R3,LUNIT        ; STORE LOGICAL UNIT NUMBER FOR DRIVE
001600  104202  000001          MOV     #UNIT0,R2       ; GET UNIT 0 INTERCONNECT CODE
001602  110601                  ROR     R1              ; DIVIDE UNITNB BY FOUR
001603  110601                  ROR     R1
001604  103201  177760          BIC     #LBLONB,R1      ; CLEAR UNUSED BITS
001606  117401          PORT3:  DEC     R1
001607  071614                  BMI     PORT4           ; FOR EACH DRIVE OVER 0 SHIFT R2 LEFT
001610  110202                  ROL     R2
001611  103202  000001          BIC     #1,R2           ; CLEAR CARRY ROTATED INTO REG (IF ANY)
001613  001606                  BR      PORT3
001614  104020  002646  PORT4:  MOV     R2,SDI          ; STORE SDI INTERCONNECT CODE
001616  001633                  BR      OVRLAY          ; BRING OVERLAY IN
                                                        ; PERFORM TEST ON THIS DRIVE
                                                        ; TEST RETURNS TO TESTX
001617  104206  001542  TESTX:  MOV     #STACK,SP       ; RESET STACK DUE TO JUMPS OUT OF
                                                        ; SUBROUTINES
001621  104301  002645          MOV     UNITNB,R1       ; GET UNIT INDEX
001623  115401          PORT5:  INC     R1              ; INCREMENT INDEX
001624  106201  000020          CMP     #16.,R1         ; CHECK IF 16 DRIVES ALREADY SELECTED
001626  051556                  BNE     PORT2           ; REPEAT FOR ALL DRIVES
001627  104207  060016  DONECD: MOV     #DONE,R0        ; END OF PROGRAM
001631  021000                  CALL    HOSTRQ
001632  001627                  BR      DONECD          ; REPEAT IF RETURNED
```

```
                                        .SBTTL  OVRLAY - OVERLAY PROCESS FOR BRINGING IN OVERLAY
                                ; *** BRING IN OVERLAY
001633  104206  001542      OVRLAY: MOV     #STACK,SP               ; RESET STACK DUE TO JUMPS OUT OF
001635  104307  001730              MOV     OVRLNM,R0               ; GET WHICH OVERLAY TO BRING IN
001637  103207  177776              BIC     #177776,R0              ; CLEAR ALL BUT LAST BIT
001641  115407                      INC     R0                      ; INCREMENT POINTER
001642  104070  001731              MOV     R0,OVRPNT               ; SAVE OVERLAY NUMBER FOR ERROR
001644  110207                      ROL     R0                      ; AND SHIFT FOR PROPER POINTER
001645  104205  000003              MOV     #3,R5                   ; GET 3 TRIES
001647  104671  001550      1$:     MOV     OTABLE+1(R0),R1         ; GET THE HI ORDER BITS OF UNIBUS ADDRESS
001651  104012                      MOV     R1,R2                   ; GET NUMBER OF WORDS OF TRANSFER
001652  110602                      ROR     R2                      ; ROTATE BIS 0 CORRECT POSITION
001653  110602                      ROR     R2
001654  103202  140000              BIC     #140000,R2              ; CLEAR UNUSED BITS
001656  103201  177774              BIC     #177774,R1              ; CLEAR UNUSED BITS FOR HI ORDER
001660  104203  005262              MOV     #OVRLPT,R3              ; POINT TO WHERE TO LOAD OVERLAY IN UDA RAM
001662  104677  001547              MOV     OTABLE(R0),R0           ; GET LO ORDER UNIBUS ADDRESS
001664  060013                      XFC     UREAD                   ; READ THE OVERLAY INTO UDA MEMORY
001665  115001                      TST     R1                      ; DONE?
001666  011724                      BEQ     2$                      ; IF SO, BRANCH
001667  117405                      DEC     R5                      ; ELSE TRY AGAIN
001670  051647                      BNE     1$                      ; IF RETRIES NOT ALL USED, BRANCH
001671                              DEVFTL  34,<OVRPNT,#SER39>              ; UNABLE TO READ FROM HOST MEMORY
001671  104200  002220  001037                                                MOV     #MS34,OUT.04
001674  104300  001731  001040                                                MOV     OVRPNT,OUT.05
001677  104200  002532  001041                                                MOV     #SER39,OUT.06
001702  104300  001475  001036                                                MOV     LUNIT,OUT.03
001705  104202  045732                                                        MOV     #34!FTLDEV+3000.,R2
001707  104020  001035                                                        MOV     R2,OUT.02
001711  104200  001711  001034                                                MOV     #.,OUT.01
001714  104200  060013  001033                                                MOV     #ERRMES,OUT.RQ
001717  104200  177777  001036              MOV     #-1,OUT.03      ; FLAG AS NOT ASSOCIATED WITH ANY UNIT
001722  021753                          CALL    TESTED          ; AND REPORT ERROR
001723  001727                          BR      3$              ; AND BRANCH
001724  115400  001730      2$:     INC     OVRLNM          ; GET READY FOR NEXT OVERLAY
001726  025262                          CALL    OVRLPT          ; GO EXECUTE CODE
001727  001617              3$:     BR      TESTX           ; HERE IF ERROR

000000              OVRLNM: .WORD   0               ; OVERLAY NUMBER
000000              OVRPNT: .WORD   0
```

```
                                                .SBTTL  ERROR EXIT
001732                                  TESTEW: CERROR  5,<#SER23,R1>           ; PRINT REAL TIME DRIVE STATE ONLY
001732  104200  003344  001040                            MOV     #SER23,OUT.05
001735  104010  001041                                    MOV     R1,OUT.06
001737  001753                                  BR      TESTED
001740                                  TESTEV: ENDERR  5
001740  104200  003276  001040                            MOV     #SER22,OUT.05
001743  104200  000006  003116                            MOV     #5+1,ERRPOS     ; SET THE POSITION
001746  104203  001033                  TESTEX: MOV     #OUT.RQ,R3              ; SET UP POSITION IN R3
001750  105303  003116                          ADD     ERRPOS,R3
001752  022131                          TESTEY: CALL    STRST                   ; GET STATUS FROM ST
001753  104302  002646                  TESTED: MOV     SDI,R2                  ; R2 HAS INTERCONNECT CODE
001755  060011                                  XFC     DINIT                   ; AND INIT DRIVE
001756  104307  001033                          MOV     OUT.RQ,R0               ; PRINT ERROR
001760  021000                                  CALL    HOSTRQ
001761  104200  000377  002741  DR.CLR: MOV     #LOBYTE,ERRORS          ; CLEAR ALL ERRORS
001764  104203  002662                          MOV     #CR.CLR,R3
001766  021076                                  CALL    TALK
001767  000000                                  RETURN                          ; GO TO CALLING ROUTINE
```

```
                                        .SBTTL  FNDCYL - FIND CYLINDER
                                ;
                                ;       FNDCYL  FIND CYLINDER
                                ;
                                ;       INPUTS: R4 -> HIGHEST CYL (VAR1)
                                ;               R5 -> DESIRED BN (VAR2)
                                ;               R3 = VAR3
                                ;               R1 = VAR4
                                ;
001770                          FNDCYL:
001770  106200  000104  003115        CMP     #'D,LETTER              ;IS IT A DIAGNOSTIC BLOCK?
001773  052006                        BNE     FND3
                                ; *** FOR DBN AREA
001774  104205  003062                MOV     #LCDBN,R5
001776  104204  003067        FNDCY2: MOV     #FDIACYL,R4
002000  103200  170000  003070        BIC     #^CHBHINB,FDIACYL+1
002003  104303  003073                MOV     SECTRK,R3
002005  002034                        BR      FND4
002006  106200  000114  003115  FND3: CMP     #'L,LETTER              ;IS IT A LOGICAL BLOCK?
002011  052025                        BNE     1$                      ;IF NOT, BRANCH
                                ; *** FOR LBN AREA
002012  114000  003076                CLR     TSTCYL
002014  114000  003077                CLR     TSTCYL+1
002016  104204  003076                MOV     #TSTCYL,R4
002020  104303  003017                MOV     SUB+LBNTRK,R3           ; R3 = LBN'S PER TRACK (VAR3)
002022  103203  177400                BIC     #HIBYTE,R3
002024  002034                        BR      FND4                    ;CONTINUE
                                ; *** FOR XBN AREA
002025  104204  003071  1$:     MOV     #FXBNCYL,R4             ;SET POINTER TO FIRST XBN CYL->VAR2
002027  103200  170000  003072        BIC     #^CHBHINB,FXBNCYL+1
002032  104303  003073                MOV     SECTRK,R3               ;R3 = SECTORS PER TRACK (VAR3)
002034  104302  003007  FND4:   MOV     SUB+LBNCYL+1,R2
002036  103202  007777                BIC     #HBHINB,R2
002040  101642  000001                BIS     1(R4),R2
002042  100642  000001                MOV     R2,1(R4)
002044                          4$:
002044  104247                        MOV     (R4)+,R0                ;START LOADING VARIABLES
002045  104070  003123                MOV     R0,VAR1
002047  104147                        MOV     (R4),R0
002050  104070  003124                MOV     R0,VAR1+1
002052  104257                        MOV     (R5)+,R0
002053  104070  003125                MOV     R0,VAR2
002055  104157                        MOV     (R5),R0
002056  104070  003126                MOV     R0,VAR2+1
002060  104030  003127                MOV     R3,VAR3
002062  104010  003130                MOV     R1,VAR4
002064  104207  003123                MOV     #VAR1,R0                ; R0 -> VARIABLES
002066  104201  003006                MOV     #SUB,R1                 ; R1 -> SUBUNIT CHARACTERISTICS
002070  060020                        XFC     CVT                     ;CONVERT TO CYLINDER VALUE
002071  106200  000104  003115        CMP     #'D,LETTER              ; DON'T COMPARE IF IN DBN AREA
002074  012117                        BEQ     7$
002075  106300  003133  003122        CMP     GROUP,OLDGRP            ; HAVE WE GONE OVER 2 CYL BOUNDARIES?
002100  032114                        BPL     6$                      ; IF NOT, BRANCH
002101  107300  003103  003074        SUB     BLOCKC,TSTBLK           ; ELSE, GO BACK ONE CYLYINDER
002104  042107                        BCC     5$                      ; IF NOT CARRY, BRANCH
002105  117400  003075                DEC     TSTBLK+1
002107  114000  003122        5$:     CLR     OLDGRP                  ; NEW GROUP
```

```
     58 002111  117404                          DEC     R4                      ; RESET POINTERS
     59 002112  117405                          DEC     R5
     60 002113  001770                          BR      FNDCYL                  ; AND TRY AGAIN
     61
     62 002114  104300  003133  003122  6$:     MOV     GROUP,OLDGRP            ; SAVE IN OLD GROUP FOR NEXT TIME THROUGH
     63 002117  104300  003131  003076  7$:     MOV     CYLLO,TSTCYL            ;STORE IN TSTCYL
     64 002122  104300  003132  003077          MOV     CYLLO+1,TSTCYL+1
     65 002125  104300  003133  003100          MOV     GROUP,TSTCYL+2
     66 002130  000000                          RETURN
```

```
                                        .SBTTL  STRST - STORE INFORMATION IN AREA 'ST'
                                ;
                                ;       STRST   STORE ST
                                ;
                                ;       INPUT   R3 -> OUTPUT BUFFER
                                ;       OUTPUT  OUTPUT BUFFER FILLED WITH VALUES FROM ST
                                ;
 002131                         STRST:  PUSH    <R0,R1,R2>
 002131  100467                                                                  MOV R0,-(SP)
 002132  100461                                                                  MOV R1,-(SP)
 002133  100462                                                                  MOV R2,-(SP)
 002134  104302  002646                 MOV     SDI,R2                 ;R2 HAS INTERCONNECT CODE
 002136  060007                         XFC     STATUS                 ;GET DRIVE STATE
 002137  103201  077674                 BIC     #077674,R1             ;CLEAR UNUSED BITS
 002141  100231                         MOV     R1,(R3)+               ; STORE IN OUT BUFFER
 002142  104202  000007                 MOV     #7,R2                  ; R2 IS # OF WORDS TO MOVE
 002144  104207  002773                 MOV     #ST+7,R0               ; R0 -> STATUS
 002146  104471                 1$:     MOV     -(R0),R1
 002147  100231                         MOV     R1,(R3)+
 002150  117402                         DEC     R2
 002151  052146                         BNE     1$
 002152                                 POP     <R2,R1,R0>
 002152  104262                                                                  MOV (SP)+,R2
 002153  104261                                                                  MOV (SP)+,R1
 002154  104267                                                                  MOV (SP)+,R0
 002155  000000                         RETURN
```

```
                                        .SBTTL  SEEK
                                ;SEEK
                                ;
                                ;SEEK TO CYLINDER POINTED TO BY CONTENTS OF R1
                                ;
                                ;INPUTS:
                                ;       R1 - POINTER TO CYLINDER NUMBER
                                ;       R2 - SDI INTERCONNECT
                                ;
002156                          SEEK:   PUSH <R0,R1,R3>
002156  100467                                                                  MOV R0,-(SP)
002157  100461                                                                  MOV R1,-(SP)
002160  100463                                                                  MOV R3,-(SP)
002161  104302  002646                 MOV     SDI,R2
002163  104210  002751                 MOV (R1)+,INS+1         ;PUT CYLINDER INTO COMMAND
002165  104210  002752                 MOV (R1)+,INS+2
002167  104110  002753                 MOV (R1),INS+3          ;PUT GROUP INTO COMMAND
002171  104203  002713         SEEKA:  MOV #CR.SEK,R3          ;POINT TO COMMAND
002173  104200  002706  002647         MOV     #MS.SEK,COMND   ; SET UP FOR ERROR
002176  021076                 CALL    TALK                    ; INITIATE SDI INTERCHANGE
002177  115002                 TST     R2                      ; SEE IF ERROR OCCURRED
002200  012203                 BEQ     SEEK1                   ; IF NOT, BRANCH
002201  021746                 CALL    TESTEX                  ; IF SO, REPORT
002202  002325                 BR      SEEK3                   ; AND EXIT
002203  114003         SEEK1:  CLR R3                          ;SET UP WORST CASE SEEK TIME
002204  104302  002646         SEEK2:  MOV     SDI,R2          ; MOVE MASK TO R2
002206  020754                 CALL    RDSTAT                  ; GET DRIVE STATUS
002207  115002                 TST     R2                      ; WAS IT OK?
002210  012260                 BEQ     2$                      ; IF SO, BRANCH
002211  102201  000400         BIT     #RCVERR,R1              ; SEE WHICH ERROR
002213  052236                 BNE     1$                      ; AND BRANCH
002214                         HARDER  10                      ; REPORT INVALID STATUS ERROR
002214  104200  000341  001037                                          MOV     #MS10,OUT.04
002217  104300  001475  001036                                          MOV     LUNIT,OUT.03
002222  104202  105702                                                  MOV     #10!ERHARD+3000.,R2
002224  104020  001035                                                  MOV     R2,OUT.02
002226  104200  002226  001034                                          MOV     #.,OUT.01
002231  104200  060013  001033                                          MOV     #ERRMES,OUT.RQ
002234  021732                 CALL    TESTEW                  ; BRANCH TO DONE
002235  002325                 BR      SEEK3
002236                  1$:
002236                         HARDER  12                      ; REPORT XMIT ERROR
002236  104200  000434  001037                                          MOV     #MS12,OUT.04
002241  104300  001475  001036                                          MOV     LUNIT,OUT.03
002244  104202  105704                                                  MOV     #12!ERHARD+3000.,R2
002246  104020  001035                                                  MOV     R2,OUT.02
002250  104200  002250  001034                                          MOV     #.,OUT.01
002253  104200  060013  001033                                          MOV     #ERRMES,OUT.RQ
002256  021732                 CALL    TESTEW                  ; BRANCH TO DONE
002257  002325                 BR      SEEK3
002260                  2$:
002260  102201  100000         BIT #RWRDY,R1                   ;IS R/W READY SET?
002262  052325                 BNE SEEK3                       ;  YES
002263  115403                 INC R3                          ;BUMP COUNT
002264  052204                 BNE SEEK2                       ;KEEP WAITING
002265                         HARDER  27,<INS+1,INS+2,INS+3>  ;SEEK COMPLETE TIME OUT
002265  104200  001633  001037                                          MOV     #MS27,OUT.04
```

```
        002270  104300  002751  001040
        002273  104300  002752  001041
        002276  104300  002753  001042
        002301  104300  001475  001036
        002304  104202  105723
        002306  104020  001035
        002310  104200  002310  001034
        002313  104200  060013  001033
    42                                  ;       MOV     OUT.RQ,R0
    43                                  ;       CALL    HOSTRQ
    44 002316                                   ENDERR  8
        002316  104200  003276  001043
        002321  104200  000011  003116
    45 002324  021746                           CALL    TESTEX
    46 002325                           SEEK3:  POP <R3,R1,R0>
        002325  104263
        002326  104261
        002327  104267
     ' 002330  000000                           RETURN
```



```
MOV     INS+1,OUT.05
MOV     INS+2,OUT.06
MOV     INS+3,OUT.07
MOV     LUNIT,OUT.03
MOV     #27!ERHARD+3000.,R2
MOV     R2,OUT.02
MOV     #.,OUT.01
MOV     #ERRMES,OUT.RQ


MOV     #SER22,OUT.08
MOV     #8+1,ERRPOS     ; SET THE POSITION


        MOV (SP)+,R3
        MOV (SP)+,R1
        MOV (SP)+,R0
```

```
                                        .SBTTL  READ1 - READ SECTORS ON A TRACK FOR TEST
                                ;READ1
                                ;
                                ;READ SECTORS ON THE SELECTED TRACK UNTIL ONE IS READ CORRECTLY
                                ;OR THE MAXIMUM NUMBER OF SECTORS IS READ
                                ;
                                ;INPUTS:
                                ;       R3 HAS MAX NUMBER OF SECTORS TO READ
                                ;       TRACK HAS TRACK NUMBER
                                ;       R5 -> POINTER TO BLOCK NUMBER OF FIRST SECTOR
                                ;OUTPUTS:
                                ;       R2 IS CLOBBERED
                                ;
002331                          READ1:  PUSH <R0,R1,R3,R4,R5>
002331  100467                                                                 MOV R0,-(SP)
002332  100461                                                                 MOV R1,-(SP)
002333  100463                                                                 MOV R3,-(SP)
002334  100464                                                                 MOV R4,-(SP)
002335  100465                                                                 MOV R5,-(SP)
002336  104304  003134                 MOV     TRACK,R4                ;R4 IS TRACK
002340  104207  003661                 MOV #RBUF0+RW.LOW,R0
002342  104251                         MOV (R5)+,R1                    ;PUT CLOCK NUMBER IN
002343  100271                         MOV R1,(R0)+                    ;  READ BUFFER
002344  104151                         MOV (R5),R1
                                ;
002345  106200  000130  003115        CMP     #'X,LETTER              ; DO WE DO PROCESS FOR XBN?
002350  052364                         BNE     9$                      ; IF NOT, BRANCH
                                ; *** FOR XBN AREA
002351  104305  003010                 MOV     SUB+HIXBN,R5            ; DO SPECIAL PROCESSING TO PUT HEADERIN PLACE
002353  110605                         ROR     R5                      ; SHIFT TO NEXT NIBBLE
002354  110605                         ROR     R5
002355  110605                         ROR     R5
002356  110605                         ROR     R5
002357  103205  170377                 BIC     #HBLONB,R5              ; STRIP OFF UNUSED PORTION
002361  101205  120000                 BIS     #HD.XBN,R5              ; SET HEADER CODE
002363  002407                         BR      10$
002364  106200  000104  003115  9$:    CMP     #'D,LETTER              ; DO WE PROCESS FOR DBN?
002367  012375                         BEQ     11$                     ; IF SO, BRANCH
                                ; *** FOR LBN AREA
002370  104305  003010                 MOV     SUB+HILBN,R5            ; GET HI LBN VALUE (DON'T HAVE TO ROR)
002372  103205  170377                 BIC     #HBLONB,R5
002374                                 ASSUME HD.LBN,0
002374  002407                         BR      10$
                                ; *** FOR DBN AREA
002375  104305  003011          11$:   MOV     SUB+HIDBN,R5            ; GET HIGH ORDER BITS OF STARTING DBN
002377  110605                         ROR     R5                      ; MOVE TO CORRECT POSITION
002400  110605                         ROR     R5
002401  110605                         ROR     R5
002402  110605                         ROR     R5
002403  103205  170377                 BIC     #HBLONB,R5              ; CLEAR UNUSED BITS
002405  101205  140000                 BIS     #HD.DBN,R5              ; SET HEADER CODE
002407  105051                 10$:    ADD     R5,R1
002410  100271                         MOV R1,(R0)+
002411  104171                         MOV (R0),R1                     ;PUT IN TRACK NUMBER
002412  103201  000377                 BIC #LOBYTE,R1                  ;(MERGE WITH REAL-TIME COMMAND)
002414  101041                         BIS R4,R1
002415  100171                         MOV R1,(R0)
```

```
     53 002416  104207  003657          READ1A: MOV #RBUF0,R0           ;POINT TO READ BUFFER
     54 002420  104302  002645                  MOV     SDI,R2          ; MOVE MASK TO R2
     55 002422  020754                          CALL    RDSTAT          ; GET DRIVE STATUS
     56 002423  115002                          TST     R2              ; WAS IT OK?
     57 002424  012474                          BEQ     2$              ; IF SO, BRANCH
     58 002425  102201  000400                  BIT     #RCVERR,R1      ; SEE IF ANY ERRORS
     59 002427  052452                          BNE     1$              ; REPORT
     60 002430                                  HARDER  10              ; REPORT INVALID STATUS ERROR
        002430  104200  000341  001037                  MOV     #MS10,OUT.04
        002433  104300  001475  001036                  MOV     LUNIT,OUT.03
        002436  104202  105702                          MOV     #10!ERHARD+3000.,R2
        002440  104020  001035                          MOV     R2,OUT.02
        002442  104200  002442  001034                  MOV     #.,OUT.01
        002445  104200  060013  001033                  MOV     #ERRMES,OUT.RQ
     61 002450  021732                          CALL    TESTEW          ; BRANCH TO DONE
     62 002451  002617                          BR      READ1X
     63 002452                          1$:
     64 002452                                  HARDER  12              ; REPORT XMIT ERROR
        002452  104200  000434  001037                  MOV     #MS12,OUT.04
        002455  104300  001475  001036                  MOV     LUNIT,OUT.03
        002460  104202  105704                          MOV     #12!ERHARD+3000.,R2
        002462  104020  001035                          MOV     R2,OUT.02
        002464  104200  002464  001034                  MOV     #.,OUT.01
        002467  104200  060013  001033                  MOV     #ERRMES,OUT.RQ
     65 002472  021732                          CALL    TESTEW          ; BRANCH TO DONE
     66 002473  002617                          BR      READ1X
     67 002474                          2$:
     68 002474  102201  100000                  BIT #RWRDY,R1           ; SEE IF READ WRITE READY IS STILL HIGH
     69 002476  052521                          BNE READ1C              ; IF SO, BRANCH
     70 002477                                  HARDER  24              ;READ/WRITE READY DROPPED BEFORE READ
        002477  104200  001306  001037                  MOV     #MS24,OUT.04
        002502  104300  001475  001036                  MOV     LUNIT,OUT.03
        002505  104202  105720                          MOV     #24!ERHARD+3000.,R2
        002507  104020  001035                          MOV     R2,OUT.02
        002511  104200  002511  001034                  MOV     #.,OUT.01
        002514  104200  060013  001033                  MOV     #ERRMES,OUT.RQ
     71 002517  021740                          CALL    TESTEV          ; BRANCH
     72 002520  002617                          BR      READ1X
     73 002521  104302  002646          READ1C: MOV     SDI,R2
     74 002523  060012                          XFC WAITSI              ;WAIT FOR SECTOR OR INDEX PULSE B4 READ
     82 002524  060002                          XFC XREAD               ;READ THE SECTOR
     83 002525  115001                          TST R1                  ;CHECK FOR ERROR
     84 002526  012617                          BEQ READ1X              ;END ROUTINE IF READ OK
     85 002527  117403                          DEC R3                  ;COUNT MAX SECTORS TO READ
     86 002530  012543                          BEQ READ1E              ;REPORT ERROR IF ALL READ
     87 002531  104207  003661                  MOV #RBUF0+RW.LOW,R0
     88 002533  104171                          MOV (R0),R1             ;INCREMENT BLOCK NUMBER
     89 002534  115401                          INC R1
     90 002535  100271                          MOV R1,(R0)+
     91 002536  042542                          BCC READ1B
     92 002537  104171                          MOV (R0),R1
     93 002540  115401                          INC R1
     94 002541  100171                          MOV R1,(R0)
     95 002542  002416          READ1B: BR READ1A                       ;GO READ NEXT SECTOR
     96 002543          READ1E: HARDER  28,<LETTER,RBUF0+RW.LOW,RBUF0+RW.HI,INS+1,INS+2,INS+3,RBUF0+RW.CMD>
        002543  104200  001716  001037                  MOV     #MS28,OUT.04
        002546  104300  003115  001040                  MOV     LETTER,OUT.05
```

```
        002551  104300  003661  001041                          MOV     RBUF0+RW.LOW,OUT.06
        002554  104300  003662  001042                          MOV     RBUF0+RW.HI,OUT.07
        002557  104300  002751  001043                          MOV     INS+1,OUT.08
        002562  104300  002752  001044                          MOV     INS+2,OUT.09
        002565  104300  002753  001045                          MOV     INS+3,OUT.10
        002570  104300  003663  001046                          MOV     RBUF0+RW.CMD,OUT.11
        002573  104300  001475  001036                          MOV     LUNIT,OUT.03
        002576  104202  105724                                  MOV     #28!ERHARD+3000.,R2
        002600  104020  001035                                  MOV     R2,OUT.02
        002602  104200  002602  001034                          MOV     #.,OUT.01
        002605  104200  060013  001033                          MOV     #ERRMES,OUT.RQ
     97 002610                          ENDERR  12
        002610  104200  003276  001047                          MOV     #SER22,OUT.12
        002613  104200  000015  003116                          MOV     #12+1,ERRPOS    ; SET THE POSITION
     98 002616  021746                  CALL    TESTEX
     99                         ;       MOV     OUT.RQ,R0
    100                         ;       CALL    HOSTRQ
    101 002617                  READ1X: POP <R5,R4,R3,R1,R0>
        002617  104265                                                      MOV (SP)+,R5
        002620  104264                                                      MOV (SP)+,R4
        002621  104263                                                      MOV (SP)+,R3
        002622  104261                                                      MOV (SP)+,R1
        002623  104267                                                      MOV (SP)+,R0
    102 002624  000000                  RETURN
```

```
                                        .SBTTL  UNITS, SDI COMMANDS AND PROGRAM VARIABLES
                                ;PROGRAM VARIABLES
                                ;UNIT NUMBER STORAGE FOR DISK DRIVES TO TEST
002625  000020                  UNITS:  .REPT   16.             ; 16 SUBUNITS (8 MAX) 4 ON EACH UNIT
                                        .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
                                        .ENDR
002625  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002626  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002627  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002630  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002631  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002632  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002633  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002634  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002635  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002636  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002637  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002640  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002641  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002642  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002643  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
002644  177777                          .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH

002645  000000                  UNITNB: .WORD 0                 ;NUMBER OF UNIT CURRENTLY UNDER TEST
                                                                ;POINTER TO TABLE ABOVE
002646  000000                  SDI:    .WORD 0                 ;SDI INTERCONNECT CODE FOR XFC CALLS
002647  000000                  COMND:  .WORD   0               ; POINTER TO COMMAND MESSAGE
                                ;       MESSAGE TABLES
002650                          CR.ONL: MSG     ONL,2,ST,7,COMPLT       ; DRIVE ONLINE
002650  002732                          .WORD ONL                       ;ADDRESS OF COMMAND
002651  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
002652  002764                          .WORD ST                        ;ADDRESS OF REPLY
002653  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
002654  000176                          .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
002655                          CR.GCR: MSG     GCR,1,CR,12.,170        ; GET CHARACTERISTICS
002655  002742                          .WORD GCR                       ;ADDRESS OF COMMAND
002656  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
002657  002773                          .WORD CR                        ;ADDRESS OF REPLY
002660  000014                          .WORD 12.                       ;SIZE OF REPLY IN WORDS
002661  000170                          .WORD 170                       ; SUCCESSFUL COMPLETION CODE
002662                          CR.CLR: MSG     DRC,2,ST,7,COMPLT       ; DRIVE CLEAR
002662  002740                          .WORD DRC                       ;ADDRESS OF COMMAND
002663  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
002664  002764                          .WORD ST                        ;ADDRESS OF REPLY
002665  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
002666  000176                          .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
002666                  SNDONE  =       .-1                     ;PREVIOUS COMMANDS CAN BE SENNT TO AN OFFLINE DRIVE
002667                          CR.DIS: MSG     DIS,2,ST,7,COMPLT       ; DISCONNECT
002667  002736                          .WORD DIS                       ;ADDRESS OF COMMAND
002670  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
002671  002764                          .WORD ST                        ;ADDRESS OF REPLY
002672  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
```

```
                002673  000176                  .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
 23             002674                  CR.SCR: MSG     SCR,2,SUB,21.,167       ; GET SUBUNIT CHARACTERISTICS
                002674  002743                  .WORD SCR                       ;ADDRESS OF COMMAND
                002675  000002                  .WORD 2                 ;SIZE OF COMMAND IN BYTES
                002676  003006                  .WORD SUB                       ;ADDRESS OF REPLY
                002677  000025                  .WORD 21.                       ;SIZE OF REPLY IN WORDS
                002700  000167                  .WORD 167                       ; SUCCESSFUL COMPLETION CODE
 24             002701                  CR.GST: MSG     GST,1,ST,7,366          ; GET STATUS
                002701  002745                  .WORD GST                       ;ADDRESS OF COMMAND
                002702  000001                  .WORD 1                 ;SIZE OF COMMAND IN BYTES
                002703  002764                  .WORD ST                        ;ADDRESS OF REPLY
                002704  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
                002705  000366                  .WORD 366                       ; SUCCESSFUL COMPLETION CODE
 25             002706                  CR.MOD: MSG     MOD,3,ST,7,COMPLT       ; MODE
                002706  002754                  .WORD MOD                       ;ADDRESS OF COMMAND
                002707  000003                  .WORD 3                 ;SIZE OF COMMAND IN BYTES
                002710  002764                  .WORD ST                        ;ADDRESS OF REPLY
                002711  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
                002712  000176                  .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
 26             002713                  CR.SEK: MSG     INS,6,ST,7,COMPLT       ; INITIATE SEEK
                002713  002750                  .WORD INS                       ;ADDRESS OF COMMAND
                002714  000006                  .WORD 6                 ;SIZE OF COMMAND IN BYTES
                002715  002764                  .WORD ST                        ;ADDRESS OF REPLY
                002716  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
                002717  000176                  .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
 27             002717                  LONG = .-1                              ; LONG TIMEOUT COMMANDS FOLLOW
 28             002720                  CR.INR: MSG     INR,1,ST,7,COMPLT       ; INITIATE RECALIBRATE
                002720  002747                  .WORD INR                       ;ADDRESS OF COMMAND
                002721  000001                  .WORD 1                 ;SIZE OF COMMAND IN BYTES
                002722  002764                  .WORD ST                        ;ADDRESS OF REPLY
                002723  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
                002724  000176                  .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
 29             002725                  CR.RUN: MSG     RUN,1,ST,7,COMPLT       ; RUN
                002725  002746                  .WORD RUN                       ;ADDRESS OF COMMAND
                002726  000001                  .WORD 1                 ;SIZE OF COMMAND IN BYTES
                002727  002764                  .WORD ST                        ;ADDRESS OF REPLY
                002730  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
                002731  000176                  .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
 30
 31                                     ;       LEVEL 2 COMMAND MESSAGE DATA STRUCTURES
 32
 33             002732     000     213  ONL:    .BYTE   0,213                   ; BRING DRIVE ONLINE
 34             002733  000012          .WORD   10.
 35             002734     000     003  DIA:    .BYTE   0,3                     ; DIAGNOSE
 36             002735  000000          .WORD   0
 37             002736     000     204  DIS:    .BYTE   0,204                   ; DISCONNECT
 38             002737  000000          .WORD   0
 39             002740     000     005  DRC:    .BYTE   0,DRVCLR                ; DRIVE CLEAR
 40             002741  000000          ERRORS: .WORD   0                       ; ERROR FLAGS
 41             002742     000     207  GCR:    .BYTE   0,GETCHR                ; GET CHARACTERISTICS
 42             002743     000     210  SCR:    .BYTE   0,GETSUB                ; GET SUBUNIT CHARACTERISTICS
 43             002744  000000          SUBUNT: .WORD   0                       ; SUBUNIT SELECTION IN LOW ORDER BYTE
 44             002745     000     011  GST:    .BYTE   0,GETSTA                ; GET STATUS
 45             002746     000     014  RUN:    .BYTE   0,14                    ; SPI  UP DRIVE
 46             002747     000     216  INR:    .BYTE   0,IRECLB                ; INITIATE RECALIBRATE
 47             002750     000     012  INS:    .BYTE   0,INSEEK                ; INITIATE SEEK
 48             002751  000000          .WORD   0                               ; INS CYLINDER/HEAD ARGUMENTS
```

```
     002752  000000                          .WORD   0
     002753  000000                          .WORD   0
     002754     000      201     MOD:        .BYTE   0,201                   ; CHANGE MODE
     002755  000000             MODE:       .WORD   0
     002756     362      377     MODE1:      .BYTE   362,377
     002757     006      377     MODE2:      .BYTE   6,377

     002760     000      014     RUNCMD:     .BYTE   0,14
             000014              RUNLBC      =       14                      ; RUN COMMAND

     002761  000000             COPY:       .WORD   0
     002762  000000             RETRY:      .WORD   0
     002763  000000             SERRTY:     .WORD   0                       ; RETRY COUNT

                                ;           RESPONSE  MESSAGE DATA BUFFERS

     002764                     ST:         .BLKW   7                       ; STATUS MESSAGE BUFFER
     002773                     CR:         .BLKW   31.                     ; CHARACTERISTICS MESSAGE BUFF
                                ;           ASSUME  SUB,CR+11.

     003032  000000             DMSDI:      .WORD   0                       ; DUMMY SDI CONTROL BLK
     003033  000000             NSCSL:      .WORD   0                       ; # OF SECTORS IN HEADER SEARCH
     003034  003001                         .WORD   SUB-5                   ; POINTER TO SUBUNIT CHAR.
     003035                              .BLKW   5                          ; WORD DMSDI+7 IS CLOBBERED BY UDA
                                                                            ; SO SET ASIDE SPACE
     003042                              .BLKW   2                          ; RESERVED FOR UDA PRIMARY REVECTORING
     003044                              .BLKW   8.                         ; SKIP SPACE TO POINTER
     003054  003026                      .WORD   DMSDI-4                    ; PRI REV INFO WRITTEN IN DMSDI+8 AND 9

                                ;           DISK LOCATION POINTERS

     003055                     ROFDBN:     .BLKW   2                       ; FIRST READ ONLY DBN
     003057                     ROFDC:      .BLKW   3                       ; FIRST READ ONLY CYL AND GROUP
     003062                     LCDBN:      .BLKW   2                       ; FIRST FACTORY FORMATTED DBN
     003064                     LDIACYL:    .BLKW   3                       ; FACTORY FORMATTED CYLINDER
     003067                     FDIACYL:    .BLKW   2                       ; FIRST DIAGNOSTIC CYLINDER
     003071                     FXBNCYL:    .BLKW   2                       ; FIRST XBN CYLINDER
     003073                     SECTRK:     .BLKW   1                       ; SECTORS PER TRACK

     003074                     TSTBLK:     .BLKW   2                       ; TEST BLOCK NUMBER
     003076                     TSTCYL:     .BLKW   3                       ; TEST CYLINDER NUMBER
     003101                     BLOCKT:     .BLKW   1                       ; BLOCKS ON CURRENT TRACK
     003102                     BLOCKG:     .BLKW   1                       ; BLOCKS ON CURRENT GROUP
     003103                     BLOCKC:     .BLKW   1                       ; BLOCKS ON CURRENT CYL
     003104                     CURBLK:     .BLKW   2                       ; CURRENT BLOCK NUMBER
     003106                     SECGRP:     .BLKW   1                       ; NUMBER OF SECTORS PER GROUP
     003107                     SECCYL:     .BLKW   1                       ; NUMBER OF SECTORS PER CYL
     003110                     CURGRP:     .BLKW   1                       ; CURRENT GROUP NUMBER
     003111  000000             CURTRK:     .WORD   0                       ; CURRENT TRACK
     003112  000000             CURPAT:     .WORD   0                       ; CURRENT PATTERN
     003113  000000             SECCNT:     .WORD   0                       ; SECTOR COUNT

     003114  000000             AREA:       .WORD   0                       ; TO STORE AREA LETTER L OR X (BN)
     003115  000000             LETTER:     .WORD   0                       ; TO STORE CURRENT AREA L, D OR X (BN)

     003116  000000             ERRPOS:     .WORD   0
     003117  000000             WFLAG:      .WORD   0                       ; FOR WRITE PROTECTION TEST
```

```
    106
    107 003120  000000                FLAG:   .WORD   0               ; TEST ALL CYLINDER FLAG
    108 003121  000000                GRPCNT: .WORD   0               ; COUNT OF CYL'S HANDLED
    109 003122  000000                OLDGRP: .WORD   0               ; HOLDS OLD GROUP VALUE FROM EVL COMPUTE
    110
    111 003123  000000  000000        VAR1:   .WORD   0,0             ; FOR CONVERT XFC
    112 003125  000000  000000        VAR2:   .WORD   0,0
    113 003127  000000                VAR3:   .WORD   0
    114 003130  000000                VAR4:   .WORD   0
    115 003131  000000  000000        CYLLO:  .WORD   0,0
    116 003133  000000                GROUP:  .WORD   0
    117 003134  000000                TRACK:  .WORD   0
    118 003135  000000                SSCTOR: .WORD   0
    119 003136  000000                INDEXS: .WORD   0
    120
    121                               ;       WRITE DATA BUFFERS
    122
    123 003137                        CHAIN:                          ; READ AND WRITE CHAIN AREA
    131 003137  000000                        .WORD   0               ; STATUS AREA
    132 003140  000000                        .WORD   0               ; BUFFER POINTER
    133 003141  000000                        .WORD   0               ; LO ORDER HEADER
    134 003142  000000                        .WORD   0               ; HI ORDER HEADER
    135 003143  000000                        .WORD   0               ; REAL TIME COMMAND AND TRACK NUMBER
     36 003144  003032                        .WORD   DMSDI           ; POINTER TO DUMMY SDI CONTROL BLOCK
    137
```

```
                                        .SBTTL  DATA PATTERNS
003145  000004          PATPTR: .WORD   4                       ; FOUR PATTERNS WILL BE TESTED
003146  003152                  .WORD   PAT4
003147  003173                  .WORD   PAT5
003150  003214                  .WORD   PAT6
003151  003235                  .WORD   PAT8
                        ;
                        ;       THE DATA PATTERNS (LENGTH OF PATTERN IN WORDS, FOLLOWED BY PATTERN)
                        ;
003152  000016          PAT4:   .WORD   16                      ; SHIFTING ONES
003153  000001                  .WORD   000001
003154  000003                  .WORD   000003
003155  000007                  .WORD   000007
003156  000017                  .WORD   000017
003157  000037                  .WORD   000037
003160  000077                  .WORD   000077
003161  000177                  .WORD   000177
003162  000377                  .WORD   000377
003163  000777                  .WORD   000777
003164  001777                  .WORD   001777
003165  003777                  .WORD   003777
003166  007777                  .WORD   007777
003167  017777                  .WORD   017777
003170  037777                  .WORD   037777
003171  077777                  .WORD   077777
003172  177777                  .WORD   177777
003173  000016          PAT5:   .WORD   16                      ; SHIFTING ZEROS
003174  177776                  .WORD   177776
003175  177774                  .WORD   177774
003176  177770                  .WORD   177770
003177  177760                  .WORD   177760
003200  177740                  .WORD   177740
003201  177700                  .WORD   177700
003202  177600                  .WORD   177600
003203  177400                  .WORD   177400
003204  177000                  .WORD   177000
003205  176000                  .WORD   176000
003206  174000                  .WORD   174000
003207  170000                  .WORD   170000
003210  160000                  .WORD   160000
003211  140000                  .WORD   140000
003212  100000                  .WORD   100000
003213  000000                  .WORD   000000
003214  000016          PAT6:   .WORD   16                      ; ALTERNATING ZERO WORD AND ONE WORD IN
003215  000000                  .WORD   000000                  ; 3-2-1-1-1 SEQUENCE
003216  000000                  .WORD   000000
003217  000000                  .WORD   000000
003220  177777                  .WORD   177777
003221  177777                  .WORD   177777
003222  177777                  .WORD   177777
003223  000000                  .WORD   000000
003224  000000                  .WORD   000000
003225  177777                  .WORD   177777
003226  177777                  .WORD   177777
003227  000000                  .WORD   000000
003230  177777                  .WORD   177777
003231  000000                  .WORD   000000
```

```
     58 003232  177777                          .WORD   177777
     59 003233  000000                          .WORD   000000
     60 003234  177777                          .WORD   177777
     61 003235  000016              PAT8:        .WORD   16              ; B'0101010101010101' AND
     62 003236  052525                          .WORD   052525          ; B'1010101010101010'
     63 003237  052525                          .WORD   052525          ; IN 3-2-1-1-1 SEQURNCE
     64 003240  052525                          .WORD   052525
     65 003241  125252                          .WORD   125252
     66 003242  125252                          .WORD   125252
     67 003243  125252                          .WORD   125252
     68 003244  052525                          .WORD   052525
     69 003245  052525                          .WORD   052525
     70 003246  125252                          .WORD   125252
     71 003247  125252                          .WORD   125252
     72 003250  052525                          .WORD   052525
     73 003251  125252                          .WORD   125252
     74 003252  052525                          .WORD   052525
     75 003253  125252                          .WORD   125252
     76 003254  052525                          .WORD   052525
     77 003255  125252                          .WORD   125252
     78
     79
     80 003256                      OBUFF:   .BLKW   257.              ; WRITE DATA BUFFER
     81         000400              BUFFSZ = 256.
     82
     83                             ;        READ DATA BUFFERS
     84
     85 003657                      RBUF0:                             ; FIRST BUFFER
     93 003657  100000                          .WORD   RSTOP           ; STATUS AND LINK POINTER
     94 003660  003665                          .WORD   RBUFD           ; POINTER TO DATA
     95 003661                                  .BLKW   2
     96 003663  013400                          .WORD   RREAL           ; LEVEL 1 SDI COMMAND
     97 003664  003032                          .WORD   DMSDI           ; POINTER TO DUMMY SDI CONTROL BLOCK
     98
```

```
                                        .SBTTL  START, GETU<POLL ALL PORTS>, RBUFD, & FCHAIN

                                .IF     GE,<.+<255.*3>>-HIMEM
                                        .ERROR
                                .ENDC

003665                          RBUFD:; .BLKW   274.                    ; READ DATA BUFFER

                                ;       FORMAT CHAIN

003665                          FCHAIN:
                                ;       .REPT   255.                    ; UP TO 240+ DBNS/TRK MAXIMUM
                                ;       .WORD   0                       ; BUFFER POINTER
                                ;       .WORD   0                       ; LO ORDER DBN
                                        .WORD   0                       ; HI ORDER DBN
                                ;       .ENDR
                                ;       .WORD   0                       ; EXTRA WORD FOR END-OF-CHAIN FLAG
003665                          START:
003665                          GETU:
                                        .SBTTL  GETU    - POLL ALL PORTS, THEN GET UNITS TO TEST
                                ;
                                ;       GET THE UNITS TO TEST
                                ;
                                ;
                                ;       POLL ALL PORTS AND FILL IN A UDA PORT INFORMATION TABLE (UNITS)
                                ;
003665  114000  001730                  CLR     OVRLNM                  ; CLEAR OVERLAY NUMBER TO INITATE

                                ; *** SET UP OVERLAY TABLE
003667  104302  007774                  MOV     OVSTRT,R2               ; GET STARTING ADDRESS OF OVERLAY (LO)
003671  104303  007775                  MOV     OVSTRT+1,R3             ; GET STARTING ADDRESS OF OVERLAY (HI)
003673  104020  001547                  MOV     R2,OTABLE               ; MOVE STARTING ADDRESS OF OVERLAY
003675  104207  000002                  MOV     #NUMOVL-1,R0            ; R0 = NUMBER OF OVERLAYS
003677  104205  001547                  MOV     #OTABLE,R5              ; R5 -> OVERLAY
003701  104651  000001          1$:     MOV     1(R5),R1                ; R1 HAS OVER LAY LENGTH
003703  110601                          ROR     R1                      ; SHIFT TO MAKE BYTES
003704  102201  100001                  BIT     #100001,R1              ; CLEAR UNUSED BITS
003706  105012                          ADD     R1,R2                   ; FIND ADDRESS OF NEXT OVERLAY
003707  043711                          BCC     2$                      ; IF NO CARRY BRANCH
003710  115403                          INC     R3                      ; PROPOGATE CARRY
003711  100652  000002          2$:     MOV     R2,2(R5)                ; STORE STARTING ADDRESS OF NEXT OVERLAY
003713  104651  000003                  MOV     3(R5),R1                ; GET OVERLAY LENGTH (<<2)
003715  101031                          BIS     R3,R1                   ; SAVE HI ORDER OVERLAY ADDRESS
003716  100651  000003                  MOV     R1,3(R5)                ; SAVE
003720  105205  000002                  ADD     #2,R5                   ; POINT TO NEXT OVERLAY AREA
003722  117407                          DEC     R0                      ; DECREMENT COUNT
003723  053701                          BNE     1$                      ; IF ALL OVERLAYS NOT SET UP, BRANCH

                                ; *** NOW DO REGULAR GETU
003724  104205  000001                  MOV     #1,R5                   ; MOVE INITIAL MASK TO R5
003726  104204  002625                  MOV     #UNITS,R4               ; R4 POINTS TO UNIT TABLE
003730                          5$:     PUSH    R4                      ; SAVE R4
003730  100464                                                                          MOV R4,-(SP)
003731  104052                          MOV     R5,R2                   ; MOVE MASK TO R2
003732  020754                          CALL    RDSTAT                  ; GET DRIVE'S STATUS
003733  115002                          TST     R2                      ; SEE IF ERROR
003734  013742                          BEQ     10$                     ; IF NOT, BRANCH
003735  104203  003052                  MOV     #SER10,R3               ; NO DRIVE ATTACHED
```

```
     63 003737  100643  000001                    MOV     R3,1(R4)        ; SAVE ERROR MESSAGE
     64 003741  004133                            BR      85$             ; REPORT
     65 003742  114003                  10$:      CLR     R3              ; SET UP TIMEOUT COUNT
     66 003743  104052                  15$:      MOV     R5,R2           ; MOVE MASK TO R2
     67 003744  020754                            CALL    RDSTAT          ; GET STATUS
     68 003745  102201  000001                    BIT     #RCVRDY,R1      ; SEE IF RECEIVER READY ASSERTED
     69 003747  053757                            BNE     20$             ; IF SO, BRANCH
     70 003750  117403                            DEC     R3              ; DECREMENT COUNT
     71 003751  053743                            BNE     15$             ; IF INCOMPLETE, BRANCH
     72 003752  104203  003065                    MOV     #SER11,R3       ; RECEIVER READY NEVER ASSERTED
     73 003754  100643  000001                    MOV     R3,1(R4)        ; SAVE ERROR MESSAGE
     74 003756  004133                            BR      85$             ; REPORT
     75 003757  102201  000100          20$:      BIT     #AVAIL,R1       ; SEE IF DRIVE IS AVAILABLE
     76 003761  053767                            BNE     25$             ; IF SO, BRANCH
     77 003762  104203  003360                    MOV     #SER40,R3       ; GET SECONDARY ERROR
     78 003764  100643  000001                    MOV     R3,1(R4)        ; SAVE
     79 003766  004133                            BR      85$             ; EXIT
     80 003767  104202  001750          25$:      MOV     #MAXSND,R2      ; SET UP MAXIMUM TRIES AT SENDING
     81 003771  104203  002701                    MOV     #CR.GST,R3      ; R3 POINTS TO GET STATUS COMMAND
     82 003773  104137                  30$:      MOV     (R3),R0         ; SET ADR OF SDI COMMAND BUFFER
     83 003774  104631  000001                    MOV     1(R3),R1        ; SET BUFFER LENGTH
     84 003776                                    PUSH    R2              ; SAVE R2
        003776  100462                                                                  MOV R2,-(SP)
     85 003777  104052                            MOV     R5,R2           ; SETUP FOR SEND
     86 004000  060004                            XFC     SEND            ; SEND COMMAND
     87 004001                                    POP     R2              ; RESTORE COUNT
        004001  104262                                                                  MOV (SP)+,R2
     88 004002  115001                            TST     R1              ; DID UNIT ACCEPT COMMAND
     89 004003  014013                            BEQ     35$             ; IF SO, BRANCH
     90 004004  117402                            DEC     R2              ; DECREMENT COUNT
     91 004005  053773                            BNE     30$             ; IF UNEXPIRED, BRANCH
     92 004006  104203  003103                    MOV     #SER12,R3       ; GET ERROR NUMBER
     93 004010  100643  000001                    MOV     R3,1(R4)        ; SAVE
     94 004012  004133                            BR      85$
     95 004013                          35$:      PUSH    R4              ; SAVE R4
        004013  100464                                                                  MOV R4,-(SP)
     96 004014  104204  000003                    MOV     #3,R4           ; SET UP SHORT TIMEOUT
     97 004016  104207  002764          40$:      MOV     #ST,R0          ; SET DATA BUFFER ADDRESS
     98 004020  104631  000003                    MOV     3(R3),R1        ; SET BUFFER LENGTH
     99 004022  104052                            MOV     R5,R2           ; SETUP FOR RECEIVE
    100 004023  060005                            XFC     RCV             ; RECEIVE SDI COMMAND
    101 004024  115001                            TST     R1              ; DID ERROR OCCUR
    102 004025  014065                            BEQ     70$             ; IF NOT, BRANCH
    103 004026  106201  000001                    CMP     #1,R1           ; SEE IF TIMEOUT
    104 004030  054037                            BNE     45$             ; IF NOT, BRANCH
    105 004031  117404                            DEC     R4              ; DECREMENT TIMEOUT VALUE
    106 004032  054016                            BNE     40$             ; IF NOT TIMEOUT, BRANCH
    107 004033                                    POP     R4              ; RESTORE R4
        004033  104264                                                                  MOV (SP)+,R4
    108 004034  104203  003115                    MOV     #SER13,R3       ; GET ERROR NUMBER
    109 004036  004062                            BR      65$             ; BRANCH TO EXIT
    110 004037                          45$:      POP     R4              ; RESTORE R4
        004037  104264                                                                  MOV (SP)+,R4
    111 004040  110601                            ROR     R1              ; ROTATE INTO POSITION TO TEST
    112 004041  110601                            ROR     R1              ; SEE IF FIRST WORD NOT START FRAME
    113 004042  044046                            BCC     50$             ; IF NOT, BRANCH
    114 004043  104203  003130                    MOV     #SER14,R3       ; GET ERROR NUMBER
```

```
    115 004045  004062                      BR      65$             ; BRANCH TO END OF LOOP
    116 004046  110601              50$:    ROR     R1              ; SEE IF FRAMING ERROR
    117 004047  044053                      BCC     55$             ; IF NOT, BRANCH
    118 004050  104203  003156              MOV     #SER15,R3       ; GET ERROR NUMBER
    119 004052  004062                      BR      65$             ; BRANCH TO END OF LOOP
    120 004053  110601              55$:    ROR     R1              ; SEE IF CHECKSUM ERROR
    121 004054  044060                      BCC     60$             ; IF NOT, BRANCH
    122 004055  104203  003200              MOV     #SER16,R3       ; GET ERROR NUMBER
    123 004057  004062                      BR      65$             ; BRANCH TO END OF LOOP
    124 004060  104203  003223      60$:    MOV     #SER17,R3       ; GET ERROR NUMBER
    125 004062  100643  000001      65$:    MOV     R3,1(R4)        ; SAVE
    126 004064  004133                      BR      85$             ; BRANCH TO END OF LOOP
```

```
                                        ;
                                        ;       NOW FILL IN THE TABLE WITH ALL THE SUBUNIT NUMBERS
                                        ;
004065                          70$:    POP     R4              ; RESTORE R4
004065  104264                                                                  MOV (SP)+,R4
004066  104207  177777                  MOV     #-1,R0          ; GET 'NO UNITS' FLAG
004070  100647  000001                  MOV     R0,1(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
004072  100647  000002                  MOV     R0,2(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
004074  104307  002764                  MOV     ST,R0           ; R0 HAS UNIT NUMBER
004076  104072                          MOV     R0,R2           ; COPY R0 TO R2
004077  103207  170000                  BIC     #^CHBHINB,R0    ; R0 HAS UNIT NUMBER
004101                                  PUSH    <R1,R2>         ; SAVE R1 AND R2
004101  100461                                                                  MOV R1,-(SP)
004102  100462                                                                  MOV R2,-(SP)
004103  104052                          MOV     R5,R2           ; MOVE UDA PORT MASK TO R2
004104  020754                          CALL    RDSTAT          ; GET STATUS
004105  102201  000002                  BIT     #ATTN,R1        ; SEE IF SPINABLE
004107  054112                          BNE     75$             ; IF SO, BRANCH
004110  101207  010000                  BIS     #10000,R0       ; SET 'NOT SPINABLE' FLAG
004112                          75$:    POP     <R2,R1>         ; RESTORE
004112  104262                                                                  MOV (SP)+,R2
004113  104261                                                                  MOV (SP)+,R1
004114  101207  040000                  BIS     #40000,R0       ; FLAG UNIT AS NOT TESTED
004116  110702                          SWAB    R2              ; SWAP R2'S BYTES
004117  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
004120  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
004121  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
004122  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
004123  103202  177760                  BIC     #LBLONB,R2      ; CLEAR ALL BUT SUBUNIT BITS
004125  100147                          MOV     R0,(R4)         ; SET UNIT NUMBER IN UNITS
004126  110602                  80$:    ROR     R2              ; MOVE SUBUNIT BIT TO CARRY
004127  044133                          BCC     85$             ; IF NO MORE SUBUNITS, BRANCH
004130  100247                          MOV     R0,(R4)+        ; MOVE SUBUNIT NUMBER TO TABLE
004131  115407                          INC     R0              ; INCREMENT SUBUNIT NUMBER
004132  004126                          BR      80$             ; BRANCH
                                        ;
                                        ;
004133                          85$:    POP     R4              ; R4 POINTS TO START OF UNIT JUST HANDLED
004133  104264                                                                  MOV (SP)+,R4
004134  105204  000004                  ADD     #4,R4           ; R4 WILL POINT TO NEXT UNIT (CLEAR CARRY FOR ROL)
004136  110205                          ROL     R5              ; R5 HAS NEXT UNIT PORT MASK
004137  106205  000C_0                  CMP     #20,R5          ; SEE IF ALL PORTS TESTED
004141  053730                          BNE     5$              ; IF NOT, BRANCH
```

```
                                ;
                                ;       NOW GET THE PLUG NUMBERS TO TEST AND FIND THEM IN THE TABLE
                                ;
004142  104207  060012          MOV     #UTOTST,R0      ; GET WHAT SUBUNIT NUMBERS TO TEST REQUEST
004144  021000                  CALL    HOSTRQ          ; GET THE PLUG NUMBERS
004145  104207  001034          MOV     #OUT.01,R0      ; R0 POINTS TO UNIT NUMBERS TO TEST
004147  104201  002625  90$:    MOV     #UNITS,R1       ; R1 POINTS TO UDA PORT INFORMATION
004151  104112          95$:    MOV     (R1),R2         ; R2 HAS UNIT
004152  074175                  BMI     115$            ; IF NONE, BRANCH
004153                          PUSH    R1              ; SAVE R1
004153  100461                                                          MOV R1,-(SP)
004154  103202  160000  100$:   BIC     #160000,R2      ; CLEAR 'NOT TESTED', LEAVE 'UNSPINABLE' SET
004156  106172                  CMP     (R0),R2         ; SEE IF IT IS A UNIT TO TEST
004157  054163                  BNE     105$            ; NO MATCH
004160  100112                  MOV     R2,(R1)         ; SAVE UNIT AS ONE TO TEST
004161                          POP     R1              ; RESTORE STACK
004161  104261                                                          MOV (SP)+,R1
004162  004302                  BR      160$            ; LOOK FOR THE NEXT ONE
004163  115401          105$:   INC     R1              ; POINT TO NEXT SUBUNIT
004164  104012                  MOV     R1,R2           ; COPY TO R2
004165  107202  002625          SUB     #UNITS,R2       ; SUBTRACT STARTING ADDRESS
004167  102202  000003          BIT     #3,R2           ; SEE IF STILL ON SAME UNIT
004171  014174                  BEQ     110$            ; IF NOT, BRANCH
004172  104112                  MOV     (R1),R2         ; SEE IF ANY MORE SUBUNITS
004173  034154                  BPL     100$            ; IF SO, BRANCH
004174          110$:   POP     R1              ; RESTORE R1
004174  104261                                                          MOV (SP)+,R1
004175  105201  000004  115$:   ADD     #4,R1           ; LOOK AT NEXT UNIT
004177  106201  002645          CMP     #UNITS+16.,R1   ; SEE IF ENTIRE TABLE SEARCHED
004201  054151                  BNE     95$             ; IF NOT, BRANCH
```

```
                                ;
                                ;       DIDN'T FIND THE REQUESTED UNITS -- DUMP ALL KNOWLEDGE OF THE
                                ;       UDA PORTS AND DIE
                                ;
 004202  104170  001036         MOV     (R0),OUT.03     ; SAVE UNIT NUMBER IN REQUEST BUFFER
 004204  104204  001040         MOV     #OUT.05,R4      ; R4 POINTS TO OUTPUT BUFFER
 004206  104205  002625         MOV     #UNITS,R5       ; R5 POINTS TO UNIT TABLE
 004210  104157                 120$:   MOV     (R5),R0         ; GET FIRST WORD OF UNIT
 004211  034216                 BPL     125$            ; IF VALID UNIT WAS FOUND, BRANCH
 004212  104657  000001         MOV     1(R5),R0        ; GET POINTER TO ERROR MESSAGE
 004214  100247                 MOV     R0,(R4)+        ; SAVE IN OUTPUT BUFFER
 004215  004251                 BR      155$            ; EXIT
 004216                         125$:   PUSH    R5              ; SAVE POINTER TO UNIT TABLE
 004216  100465                                                                 MOV R5,-(SP)
 004217  102207  010000         BIT     #10000,R0       ; SEE IF DRIVE UNSPINABLE
 004221  014225                 BEQ     130$            ; IF SPINABLE, BRANCH
 004222  104207  003402         MOV     #SER41,R0       ; REPORT DRIVE(S) UNSPINABLE
 004224  004227                 BR      135$            ; BRANCH
 004225  104207  003255         130$:   MOV     #SER18,R0       ; GET POINTER TO ERROR MESSAGE
 004227  100247                 135$:   MOV     R0,(R4)+        ; SAVE
 004230  114007                 CLR     R0              ; CLEAR COUNT
 004231  104251                 140$:   MOV     (R5)+,R1        ; GET UNIT NUMBER
 004232  074237                 BMI     145$            ; IF INVALID, BRANCH
 004233  115407                 INC     R0              ; INCREMENT COUNT
 004234  106207  000004         CMP     #4,R0           ; SEE IF MAX
 004236  054231                 BNE     140$            ; IF NOT, LOOP
 004237  104671  001542         145$:   MOV     SER18E-1(R0),R1 ; GET POINTER TO CORRECT ERROR MESSAGE
 004241  100241                 MOV     R1,(R4)+        ; MOVE INTO OUTPUT BUFFER
 004242                         POP     R5              ; RESTORE R5
 004242  104265                                                                 MOV (SP)+,R5
 004243                         PUSH    R5              ; SAVE R5
 004243  100465                                                                 MOV R5,-(SP)
 004244  104251                 150$:   MOV     (R5)+,R1        ; GET SUBUNIT NUMBER
 004245  100241                 MOV     R1,(R4)+        ; SAVE
 004246  117407                 DEC     R0              ; DECREMENT COUNT
 004247  054244                 BNE     150$            ; IF COUNT INCOMPLETE, BRANCH
 004250                         POP     R5              ; RESTORE R5
 004250  104265                                                                 MOV (SP)+,R5
 004251  105205  000004         155$:   ADD     #4,R5           ; POINT TO NEXT UNIT TABLE
 004253  106205  002645         CMP     #UNITS+16.,R5   ; SEE IF ENTIRE TABLE SEARCHED
 004255  054210                 BNE     120$            ; IF NOT, BRANCH
 004256                         DEVFTL  2000            ; REPORT FATAL ERROR (WILL SHOW UP AS A 5000)
 004256  104200  002734  001037                                 MOV     #MS2000,OUT.04
 004261  104300  001475  001036                                 MOV     LUNIT,OUT.03
 004264  104202  051610                                         MOV     #2000!FTLDEV+3000.,R2
 004266  104020  001035                                         MOV     R2,OUT.02
 004270  104200  004270  001034                                 MOV     #.,OUT.01
 004273  104200  060013  001033                                 MOV     #ERRMES,OUT.RQ
 004276  104307  001033         MOV     OUT.RQ,R0       ; SET UP FOR REPORT
 004300  021000                 CALL    HOSTRQ          ; REPORT ERROR
 004301  001627                 BR      DONECD          ; END TEST

 004302  115407                 160$:   INC     R0              ; POINT TO NEXT UNIT TO TEST
 004303  104172                 MOV     (R0),R2         ; CHECK NEXT UNIT
 004304  034147                 BPL     90$             ; FIND IN UDA PORT INFORMATION
 004305  001555                 BR      T3STRT          ; START TEST
 004306                 ENDGTU:
```

```
                                        .SBTTL  REST OF FORMAT CHAIN

004306                                  .BLKW   <<255.*3>-<ENDGTU-GETU>>

        005262                          ENDPNT = .

                                ;       OVERLAYS START HERE!!!

        005262                  OVRLPT  =       .
```

```
      1 005262                                        MESSAGES
        005262  000105                                .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
        000000     042     124     111  MS1:        .ASCII\'TIME-OUT ON SEND'N\
        000011     122     061     122              .ASCII\R1R1\
        000013     000                              .BYTE 0
        000014     042     124     111  MS2:        .ASCII\'TIME-OUT ON RECEIVE'N\
        000027     122     061     122              .ASCII\R1R1\
        000031     000                              .BYTE 0
        000032     042     106     111  MS3:        .ASCII\'FIRST WORD RECEIVED WAS NOT A START FRAME'N\
        000060     122     061     122              .ASCII\R1R1\
        000062     000                              .BYTE 0
        000063     042     106     122  MS4:        .ASCII\'FRAMING ERROR ON LEVEL 0 RESPONSE'N\
        000105     122     061     122              .ASCII\R1R1\
        000107     000                              .BYTE 0
        000110     042     103     110  MS5:        .ASCII\'CHECKSUM ERROR ON LEVEL 0 RESPONSE'N\
        000132     122     061     122              .ASCII\R1R1\
        000134     000                              .BYTE 0
        000135     042     122     105  MS6:        .ASCII\'RESPONSE LONGER THAN EXPECTED'N\
        000155     122     061     122              .ASCII\R1R1\
        000157     000                              .BYTE 0
        000160     042     103     117  MS7:        .ASCII\'CODE FROM RECEIVE WAS UNINTELLIGIBLE FROM SUBSYSTEM = 'H16N\
        000216     122     061     122              .ASCII\R1R1\
        000220     000                              .BYTE 0
        000221     042     103     117  MS8:        .ASCII\'COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
        000251     122     061                      .ASCII\R1\
        000252     042     040     040              .ASCII\'  EXPECTED RESPONSE  'H8N\
        000267     042     040     040              .ASCII\'  ACTUAL RESPONSE    'H8N\
        000304     122     061                      .ASCII\R1\
        000305     000                              .BYTE 0
        000306     042     104     122  MS9:        .ASCII\'DRIVE NOT ASSERTING RECEIVER READY I:: DRIVE STATE'N\
        000340     000                              .BYTE 0
        000341     042     106     101  MS10:       .ASCII\'FAILED TO RECEIVE VALID DRIVE STATE'NR1\
        000365     000                              .BYTE 0
        000366     042     103     101  MS11:       .ASCII\'CANNOT RECEIVE DRIVE STATE FROM DRIVE'N\
        000412     042     103     110              .ASCII\'CHECK IF DRIVE IS POWERED ON.'NR1\
        000433     000                              .BYTE 0
        000434     042     104     122  MS12:       .ASCII\'DRIVE STATE RECEIVED HAS BAD PARITY'NR1\
        000460     000                              .BYTE 0
        000461     042     116     117  MS13:       .ASCII\'NO VALID STATE FROM DRIVE'NR1\
        000500     000                              .BYTE 0
        000501     042     123     125  MS14:       .ASCII  \'SUBUNIT CHARACTERISTICS SAY THERE ARE ZERO READ ONLY GROUPS'N\
        000540     042     111     116              .ASCII  \'IN THE DIAGNOSTIC AREA'N\
        000554     000                              .BYTE   0
        000555     042     123     125  MS15:       .ASCII  \'SUBUNIT CHARACTERISTICS SAY THERE ARE LESS THAN 1 READ/WRITE'N\
        000614     042     107     122              .ASCII  \'GROUPS IN THE DIAGNOSTIC AREA'N\
        000634     000                              .BYTE   0
        000635     042     116     105  MS16:       .ASCII\'NEITHER R/W READY NOR ATTENTION SET AFTER RECALIBRATE COMMAND'NR1\
        000676     000                              .BYTE 0
        000677     042     123     125  MS17:       .ASCII\'SUBUNIT CHARACTERISTICS SAY LESS THAN 1 DIAGNOSTIC CYLINDER'N\
        000736     000                              .BYTE 0
        000737     042     122     105  MS18:       .ASCII\'READ/WRITE READY DROPPED BEFORE FORMAT OPERATION'N\
        000770     000                              .BYTE 0
        000771     042     106     117  MS19:       .ASCII\'FORMAT OPERATION REPORTED TIME-OUT FAILURE'N\
        001017     042     040     040              .ASCII\'  CYLINDER 'D28'.   GROUP  'D8'.  TRACK 'D8'.'N\
        001047     000                              .BYTE 0
        001050     042     101     106  MS20:       .ASCII\'AFTER RECAL, ERROR BITS WERE SET'NR1\
        001072     000                              .BYTE 0
```

```
001073  116  042  114  MS21:  .ASCII\N'LOGGABLE INFORMATION AFTER RECAL'NR1\
001116  000                  .BYTE 0
001117  042  122  105  MS22:  .ASCII\'READ/WRITE READY DROPPED BEFORE WRITE OPERATION'N\
001150  000                  .BYTE 0
001151  042  103  117  MS23:  .ASCII\'COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
001212  042  127  122         .ASCII\'WRITE OPERATION REPORTED FAILURE -- ERROR CODE 'O8'' OCTAL.'N\
001251  042  104  102         .ASCII\'DBN 'D24''.  CYLINDER 'D28''.  GROUP 'D8''.  TRACK 'D8''.'N\
001305  000                  .BYTE 0
001306  042  122  105  MS24:  .ASCII\'READ/WRITE READY DROPPED BEFORE READ OPERATION'N\
001336  000                  .BYTE 0
001337  042  103  117  MS25:  .ASCII\'COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
001400  042  122  105         .ASCII\'READ OPERATION REPORTED FAILURE -- ERROR CODE 'O8'' OCTAL.'N\
001436  042  104  102         .ASCII\'DBN 'D24''.  CYLINDER 'D28''.  GROUP 'D8''.  TRACK 'D8''.'N\
001472  000                  .BYTE 0
001473  042  103  117  MS26:  .ASCII\'COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
001534  042  104  101         .ASCII\'DATA COMPARE FAILURE ON WORD 'D16''.'N\
001557  042  105  130         .ASCII\'EXPECTED DATA  'H16N\
001572  042  101  103         .ASCII\'ACTUAL DATA    'H16N\
001604  042  103  131         .ASCII\'CYLINDER 'D28''.  GROUP 'D8''.  TRACK 'D8''.'N\
001632  000                  .BYTE 0
001633  042  123  105  MS27:  .ASCII\'SEEK COMPLETE TIME-OUT -- READ/WRITE READY DID NOT SET.'N\
001670  042  123  105         .ASCII\'SEEK WAS TO CYLINDER 'D28''.  GROUP 'D8''.'N\
001715  000                  .BYTE 0
001716  042  116  117  MS28:  .ASCII\'NO BLOCK ON THIS TRACK CAN BE READ. LAST BLOCK TRIED:'N\
001752  101  061  042         .ASCII\A1'BN 'D24''.  CYLINDER 'D28''.  GROUP 'D8''.  TRACK 'D8''.'N\
002006  000                  .BYTE 0
002007  042  101  126  MS29:  .ASCII  \'AVAILABLE WAS NOT ASSERTED AFTER DISCONNECT'N\
002036  042  040  040         .ASCII  \'  STATE RECEIVED  'H16N\
002052  000                  .BYTE   0
002053  042  111  116  MS30:  .ASCII\'INVALID COMMAND 'H16'' WAS SUCCESSFUL'N\
002076  000                  .BYTE   0
002077  042  103  117  MS31:  .ASCII\'COMMAND WITH 'R1'' LENGTH = 'D8'' WAS SUCCESSFUL'N\
002127  000                  .BYTE   0
002130  042  125  116  MS32:  .ASCII\'UNIT DID NOT REPORT TRANSMITTION ERROR'NR1\
002155  000                  .BYTE   0
002156  042  125  116  MS33:  .ASCII\'UNIT ACCEPTED AN INVALID GROUP NUMBER FROM GROUP SELECT LEVEL 1'N\
002217  000                  .BYTE   0
002220  042  125  116  MS34:  .ASCII\'UNABLE TO CORRECTLY READ OVERLAY 'O3NR1\
002244  000                  .BYTE   0
002245  042  123  125  MS35:  .ASCII\'SUCCESSFULLY WROTE IN DBN AREA WHEN DRIVE WAS WRITE PROTECTED'N\
002305  000                  .BYTE   0
002306  042  104  122  MS36:  .ASCII  \'DRIVE IS NOT PROPERLY FORMATTED.'NR1\
002330  000                  .BYTE   0
002331  042  104  122  MS37:  .ASCII  \'DRIVE IS FORMATTED IN 576 BYTE MODE.'N\
002354  042  124  117         .ASCII  \'TO RUN WITH A UDA, THIS DRIVE NEEDS TO BE FORMATTED '\
002407  042  111  116         .ASCII  \'IN 512 BYTE MODE.'N\
002421  000                  .BYTE   0
002422  042  116  117  MS38:  .ASCII  \'NO COPY OF THE FCT COULD BE READ.'NR1\
002445  000                  .BYTE   0
002446  042  125  104  SER36: .ASCII  \'UDA WILL SPIN DOWN THIS DRIVE IF USED IN NORMAL SYSTEM OPERATION.'N\
002510  042  124  110         .ASCII  \'THIS DRIVE NEEDS TO BE FORMATTED.'\
002531  000                  .BYTE   0
002532  042  124  110  SER39: .ASCII\'THIS UDA AND ALL DRIVES ATTACHED WILL BE REMOVED FROM TESTING'N\
002572  000                  .BYTE   0
002573  042  103  117  SER00: .ASCII\'COMMAND WAS 'R1\
002603  000                  .BYTE   0
002604  042  117  116  MS.ONL: .ASCII\'ONLINE'N\
```

```
002610  000                             .BYTE   0
002611  042  104  122  MS.CLR:  .ASCII\'DRIVE CLEAR'N\
002620  000                             .BYTE   0
002621  042  104  111  MS.DIS:  .ASCII\'DISCONNECT'N\
002627  000                             .BYTE   0
002630  042  107  105  MS.GCR:  .ASCII\'GET COMMON CHARACTERISTICS'N\
002646  000                             .BYTE   0
002647  042  107  105  MS.SCR:  .ASCII\'GET SUBUNIT CHARACTERISTICS'N\
002666  000                             .BYTE   0
002667  042  107  105  MS.GST:  .ASCII\'GET STATUS'N\
002675  000                             .BYTE   0
002676  042  103  110  MS.MOD:  .ASCII\'CHANGE MODE'N\
002705  000                             .BYTE   0
002706  042  123  105  MS.SEK:  .ASCII\'SEEK'N\
002711  000                             .BYTE   0
002712  042  111  116  MS.INR:  .ASCII\'INITIATE RECALIBRATE'N\
002725  000                             .BYTE   0
002726  042  123  120  MS.RUN:  .ASCII\'SPIN UP'N\
002733  000                             .BYTE   0
002734  042  125  116  MS2000:  .ASCII  \'UNABLE FIND REQUESTED DRIVE FOR TESTING'N\
002761  042  124  110           .ASCII  \'THE FOLLOWING IS VISIBLE ON THE PORTS'N\
003005  042  125  104           .ASCII  \'UDA PORT 0 -- 'R1\
003016  042  125  104           .ASCII  \'UDA PORT 1 -- 'R1\
003027  042  125  104           .ASCII  \'UDA PORT 2 -- 'R1\
003040  042  125  104           .ASCII  \'UDA PORT 3 -- 'R1\
003051  000                             .BYTE   0
003052  042  116  117  SER10:   .ASCII  \'NO DRIVE ATTACHED'N\
003064  000                             .BYTE   0
003065  042  122  103  SER11:   .ASCII  \'RCVR RDY NEVER ASSERTED'N\
003102  000                             .BYTE   0
003103  042  124  111  SER12:   .ASCII  \'TIMEOUT OF SEND'N\
003114  000                             .BYTE   0
003115  042  124  111  SER13:   .ASCII  \'TIMEOUT OF RECEIVE'N\
003127  000                             .BYTE   0
003130  042  106  111  SER14:   .ASCII  \'FIRST WORD RECEIVED WAS NOT START FRAME'N\
003155  000                             .BYTE   0
003156  042  106  122  SEP15:   .ASCII  \'FRAMING ERROR ON LEVEL 0 RECEIVE'N\
003177  000                             .BYTE   0
003200  042  103  110  SER16:   .ASCII  \'CHECKSUM ERROR ON LEVEL 0 RECEIVE'N\
003222  000                             .BYTE   0
003223  042  122  105  SER17:   .ASCII  \'RESPONSE LONGER THAN EXPECTED FOR GET STATUS CMD'N\
003254  000                             .BYTE   0
003255  042  104  122  SER18:   .ASCII  \'DRIVE 'R1\
003262  000                             .BYTE   0
003263  104  066  042  SER18D:  .ASCII  \D6", '\
003266  104  066  042  SER18C:  .ASCII  \D6", '\
003271  104  066  042  SER18B:  .ASCII  \D6", '\
003274  104  066  116  SER18A:  .ASCII  \D6N\
003275  000                             .BYTE   0
003276  042  122  105  SER22:   .ASCII\'REAL TIME STATE 'H16N\
003311  042  123  124           .ASCII\'STATUS (R TO L): 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
003343  000                             .BYTE   0
003344  042  122  105  SER23:   .ASCII\'REAL TIME STATE 'H16N\
003357  000                             .BYTE   0
003360  042  104  122  SER40:   .ASCII  \'DRIVE NOT AVAILABLE TO THIS UDA'N\
003401  000                             .BYTE   0
003402  042  104  122  SER41:   .ASCII  \'DRIVE NOT SPINABLE'N\
```

```
003414  000                             .BYTE   0
003415  042  103  117  SER50:  .ASCII\''COMMAND''\
003421  000                             .BYTE   0
003422  042  122  105  SER51:  .ASCII\''RESPONSE''\
003427  000                             .BYTE   0
003430  042  127  110  SER52:  .ASCII\''WHEN A CONTINUE OR END FRAME DID NOT FOLLOW A START FRAME'N\
003466  000                             .BYTE   0
003467  042  127  110  SER53:  .ASCII\''WHEN AN END FRAME WAS SENT WITH NO START FRAME'N\
003517  000                             .BYTE   0
003520  042  127  110  SER54:  .ASCII\''WHEN AN END FRAME WITH A BAD CHECKSUM WAS SENT'N\
003550  000                             .BYTE   0
003551  042  127  110  SER55:  .ASCII\''WHEN A CONTINUE FRAME WAS SENT WITH NO START FRAME'N\
003603  000                             .BYTE   0
003604  042  127  110  SER56:  .ASCII\''WHEN TWO CONSECUTIVE START FRAMES WERE SENT'N\
003633  000                             .BYTE   0
003634  042  127  110  SER57:  .ASCII\''WHEN AN END FRAME WAS SENT AFTER A START FRAME TIMED OUT'N\
003671  000                             .BYTE   0
```

```
                                .SBTTL  ***** OVERLAY 1  *****  SETUP ( AND START TESTING )
     2 003672                   DMOVLY  SU,OVRLPT
       003672  000105          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
     3                         ;*******************************************************************************
     4                         ;*******************************************************************************
     5                         ;*******************************************************************************
     6                         ;*******************************************************************************
     7          000001         SETUP   =       1
```

```
                                .SBTTL  INIT DRIVE AND LOOK AT DRIVE SIGNALS

                        ;START OF TEST CODE
                        ;
                        ;INPUTS:
                        ;       LUNIT - LOGICAL UNIT NUMBER OF DRIVE UNDER TEST
                        ;       SDI - SDI INTERCONNECT CODE FOR DRIVE

                        ;INITIALIZE THE DRIVE

005262  104302  002646  STRTST: MOV SDI,R2                      ;GET SDI SELECT CODE
005264  060011                  XFC DINIT                       ;INITIALIZE THE DRIVE

                        ;WAIT FOR DRIVE TO ASSERT RECEIVER READY
                        ;TIME OUT AFTER ...?

005265  104201  001400          MOV     #1400,R1                ; SETUP TIMEOUT
005267  117401          1$:     DEC     R1                      ; DECREMENT DELAY
005270  055267                  BNE     1$                      ; LOOP UNTIL DONE
005271  114005                  CLR     R5                      ; GET TIMEOUT COUNTER
005272  104201  000310  2$:     MOV     #200.,R1                ; INNER LOOP TIMEOUT
005274  117401          3$:     DEC     R1                      ; DELAY
005275  055274                  BNE     3$                      ;  UNTIL DONE
005276  020720                  CALL    RTDS                    ; GET REAL TIME DRIVE STATE
005277  115002                  TST     R2                      ; SEE IF ERROR OCCURRED
005300  015304                  BEQ     4$                      ; IF OK, CONTINUE
005301                          PUSH    R1
005301  100461                                                  MOV R1,-(SP)
005302  021732                  CALL    TESTEW                  ; IF SO, BRANCH
005303                          POP     R1
005303  104261                                                  MOV (SP)+,R1
005304  102201  000001  4$:     BIT     #RCVRDY,R1              ; CHECK RECEIVER READY LINES
005306  055332                  BNE     T                       ; ADVANCE IF ASSERTED
005307  117405                  DEC     R5                      ; DECREMENT TIME OUT COUNTER
005310  055272                  BNE     2$                      ; IF UNEXPIRED, BRANCH
005311                          HARDER  9                       ;REPORT ERROR
005311  104200  000306  001037          MOV     #MS9,OUT.04
005314  104300  001475  001036          MOV     LUNIT,OUT.03
005317  104202  105701                  MOV     #9!ERHARD+3000.,R2
005321  104020  001035                  MOV     R2,OUT.02
005323  104200  005323  001034          MOV     #.,OUT.01
005326  104200  060013  001033          MOV     #ERRMES,OUT.RQ
005331  021740                  CALL    TESTEV                  ;EXIT TESTING THIS DRIVE
```

```
                                        .SBTTL  GET DRIVE CHARACTERISTICS
                                ;GET DRIVE CHARACTERISTICS
                                
005332  104200  000003  001461  T:      MOV     #3,SDISTO               ; SET UP TEMPORARY SHORT TIMEOUT VALUE
005335  104203  002655                  MOV     #CR.GCR,R3              ; POINT TO GET CHARS COMMAND
005337  104200  002630  002647          MOV     #MS.GCR,COMND
005342  021076                          CALL    TALK                    ; SDI INTERCHANGE
005343  115002                          TST     R2                      ; SEE IF ERROR OCCURRED
005344  015346                          BEQ     1$                      ; IF NOT, BRANCH
005345  021746                          CALL    TESTEX                  ; IF SO, REPORT
005346  104307  002773          1$:     MOV     CR+SHRTTO,R0            ; GET SHORT TIMEOUT
005350  103207  177760                  BIC     #LBLONB,R0              ; CLEAR UNUSED BITS
005352  021463                          CALL    TO                      ; SET UP TIMEOUT
005353  104070  001461                  MOV     R0,SDISTO               ; SAVE IN SHORT TIMEOUT
005355  104307  002774                  MOV     CR+LONGTO,R0            ; GET LONG TIMEOUT
005357  103207  177760                  BIC     #LBLONB,R0              ; CLEAR UNUSED BITS
005361  021463                          CALL    TO                      ; SET UP TIMEOUT
005362  104070  001462                  MOV     R0,SDILTO               ; SAVE IN LONG TIMEOUT
005364  104307  002774                  MOV     CR+RCTCPS,R0
005366  110707                          SWAB    R0
005367  103207  177760                  BIC     #177760,R0
005371  104070  002761                  MOV     R0,COPY                 ; GET AND SAVE NUMBER OF RCT COPIES
                
                
                
                
                
                                        .SBTTL  ISSUE ONLINE COMMAND
                                ;ISSUE ONLINE COMMAND AND CHECK RESPONSE
                
005373  104203  002650                  MOV     #CR.ONL,R3              ; POINT TO ONLINE COMMAND
005375  104200  002604  002647          MOV     #MS.ONL,COMND           ; SET UP FOR ERROR
005400  021076                          CALL    TALK                    ; SDI INTERCHANGE
005401  115002                          TST     R2                      ; SEE IF ERROR OCCURRED
005402  015404                          BEQ     2$                      ; IF NOT, BRANCH
005403  021746                          CALL    TESTEX                  ; IF SO, REPORT
005404                          2$:
```

ISSUE ONLINE COMMAND

```
                                        .SBTTL  ISSUE ONLINE COMMAND
                                ;ISSUE GET STATUS COMMAND AND CHECK THAT ATTENTION DROPS

005404  104203  002701                 MOV     #CR.GST,R3              ; POINT TO GET STATUS COMMAND
005406  104200  002667  002647         MOV     #MS.GST,COMND           ; SET UP FOR ERROR
005411  021076                         CALL    TALK                    ; SDI INTERCHANGE
005412  115002                         TST     R2                      ; SEE IF ERROR OCCURRED
005413  015415                         BEQ     TOA                     ; IF NOT, BRANCH
005414  021746                         CALL    TESTEX                  ; IF SO, REPORT
005415  020720                 TOA:    CALL    RTDS                    ; GET REAL TIME DRIVE STATE
005416  102201  000002                 BIT     #ATTN,R1                ; IS ATTENTION ASSERTED?
005420  015422                         BEQ     TOB                     ; COMMAND SHOULD HAVE CLEARED IT
                                ;       ERROR - ATTEN NOT DEASSERTED
005421  021732                         CALL    TESTEW
005422                          TOB:


                                        .SBTTL  ISSUE DRIVE CLEAR COMMAND
                                ;ISSUE DRIVE CLEAR COMMAND

005422  104200  000377  002741         MOV     #LOBYTE,ERRORS          ; MOVE TO DRIVE CLEAR SDI AREA
005425  104203  002662                 MOV     #CR.CLR,R3              ; POINT TO DRIVE CLEAR
005427  104200  002611  002647         MOV     #MS.CLR,COMND           ; SET UP FOR ERROR
005432  021076                         CALL    TALK                    ; SDI INTERCHANGE
005433  115002                         TST     R2                      ; SEE IF ERROR OCCURRED
005434  015436                         BEQ     1$                      ; IF NOT, BRANCH
005435  021746                         CALL    TESTEX                  ; IF SO, REPORT
```

```
ISSUE CHANGE MODE COMMAND

                                                .SBTTL  ISSUE CHANGE MODE COMMAND
                                        ;       ISSUE CHANGE MODE COMMAND TO ENABLE DIAG CYL ACCESS
005436                                  1$:
005436  104203  002706                        MOV     #CR.MOD,R3              ; POINT TO CHANGE MODE COMMAND
005440  104300  002756  002755                MOV     MODE1,MODE              ; WRITE PROTECT, DIAG CYL ACCESS, NO FORMATTING
005443  104200  002676  002647                MOV     #MS.MOD,COMND           ; SET UP FOR ERROR
005446  021076                                CALL    TALK                    ; SDI INTERCHANGE
005447  115002                                TST     R2                      ; SEE IF ERROR OCCURRED
005450  015452                                BEQ     2$                      ; IF NOT, BRANCH
005451  021746                                CALL    TESTEX                  ; IF SO, REPOR
                                        .SBTTL  SPIN UP DRIVE
                                        ;SPIN UP DRIVE
005452  104203  002725                2$:     MOV     #CR.RUN,R3              ; POINT TO RUN COMMAND
005454  104200  002726  002647                MOV     #MS.RUN,COMND           ; SET UP FOR ERROR
005457  021076                                CALL    TALK                    ; SDI INTERCHANGE
005460  115002                                TST     R2                      ; SEE IF ERROR OCCURRED
005461  015463                                BEQ     3$                      ; IF NOT, BRANCH
005462  021746                                CALL    TESTEX                  ; IF SO, REPORT
005463                                  3$:
```

```
                                        .SBTTL  ISSUE INITIATE RECALIBRATE COMMAND
                                ; ISSUE INITIATE RECALIBRATE COMMAND
                                ;
005463  104203  002720                  MOV     #CR.INR,R3              ; POINT TO RECALIBRATE COMMAND
005465  104200  002712  002647          MOV     #MS.INR,COMND           ; SET UP FOR ERROR
005470  021076                          CALL    TALK                    ; SDI INTERCHANGE
005471  115002                          TST     R2                      ; SEE IF ERROR OCCURRED
005472  015476                          BEQ     4$                      ; IF NOT, BRANCH
005473                                  PUSH    R1
005473  100461                                                                  MOV R1,-(SP)
005474  021746                          CALL    TESTEX
005475                                  POP     R1
005475  104261                                                                  MOV (SP)+,R1
005476  104204  000004          4$:     MOV     #4,R4                   ; R4 IS DECREMENT COUNTER
005500  020720                  T3C:    CALL    RTDS                    ; GET DRIVE SIGNALS
005501  115002                          TST     R2                      ; SEE IF ERROR
005502  051732                          BNE     TESTEW                  ; IF SO, BRANCH
005503  102201  000002                  BIT     #ATTN,R1                ; DID ATTENTION SET?
005505  015604                          BEQ T3D                         ;  NO
                                ; CHECK ERROR BITS - IF SET, REPORT "AFTER RECAL ERROR BITS ARE SET" OR SOMETHING THEN EXIT
                                ; IF LOGGABLE INFO BIT SET (NO FRROR BITS) MESSAGE- LOGGABLE INFO AFTER RECAL, CONTINUE TEST
005506                                  PUSH    R1                      ; SAVE R1
005506  100461                                                                  MOV R1,-(SP)
005507  104203  002701                  MOV     #CR.GST,R3              ; POINT TO GET STATUS COMMAND
005511  104200  002667  002647          MOV     #MS.GST,COMND           ; SET UP FOR ERROR
005514  021076                          CALL    TALK                    ; SEND AND RECEIVE COMMAND
005515  115002                          TST     R2                      ; ANY ERRORS?
005516  055547                          BNE     5$                      ; IF SO, EXIT
005517  104303  002766                  MOV     ST+ST.ERR,R3            ; GET ERROR INFORMATION
005521  102203  000350                  BIT     #<ST.FE+ST.RE+ST.PE+ST.WE>,R3 ; ANY ERRORS?
005523  015554                          BEQ     T3A                     ; IF OK CONTINUE
005524                                  HARDER  20,#SER22
005524  104200  001050  001037                  MOV     #MS20,OUT.04
005527  104200  003276  001040                  MOV     #SER22,OUT.05
005532  104300  001475  001036                  MOV     LUNIT,OUT.03
005535  104202  105714                          MOV     #20!ERHARD+3000.,R2
005537  104020  001035                          MOV     R2,OUT.02
005541  104200  005541  001034                  MOV     #.,OUT.01
005544  104200  060013  001033                  MOV     #ERRMES,OUT.RQ
005547                          5$:     POP     R1
005547  104261                                                                  MOV (SP)+,R1
005550  104203  001041                  MOV     #OUT.06,R3              ; ELSE SET UP ERROR INFO
005552  021752                          CALL    TESTEY
005553  005632                          BR      T4A
005554  102200  000010  002765  T3A:    BIT     #ST.EL,ST+1             ; ANY LUGGABLE INFO?
005557  015603                          BEQ     T3B                     ; IF NOT CONTINUE
005560  104203  001037                  MOV     #OUT.04,R3              ; IF SO, SET UP POINTER
005562  022131                          CALL    STRST                   ; AND SET UP INFO
005563                                  MSSG    21,#SER22
005563  104200  001073  001035                  MOV     #MS21,OUT.02
005566  104200  003276  001036                  MOV     #SER22,OUT.03
005571  100467                                                                  MOV R0,-(SP)
005572  100461                                                                  MOV R1,-(SP)
005573  104300  001475  001034                  MOV     LUNIT,OUT.01
005576  104207  060015                          MOV     #MESSAG,R0
005600  021000                                  CALL    HOSTRQ
005601  104261                                                                  MOV (SP)+,R1
```

ISSUE INITIATE RECALIBRATE COMMAND

```
                                                                                    MOV (SP)+,R0
    005602  104267
 39 005603                          T3B:    POP     R1          ; RESTORE R1
    005603  104261                                              MOV (SP)+,R1
 40 005604  102201  100000          T3D:    BIT     #RWRDY,R1   ; DID R/W READY SET?
 41 005606  055632                          BNE T4A             ; IF SO, BRANCH
 42 005607  117404                          DEC     R4          ; TRY AGAIN?
 43 005610  055500                          BNE     T3C
 44 005611                          ;R/W READY DID NOT SET AFTER RECALIBRATE COMMAND
    005611  104200  000635  001037          MOV     #MS16,OUT.04
    005614  104300  001475  001036          MOV     LUNIT,OUT.03
    005617  104202  105710                  MOV     #16!ERHARD+3000.,R2
    005621  104020  001035                  MOV     R2,OUT.02
    005623  104200  005623  001034          MOV     #.,OUT.01
    005626  104200  060013  001033          MOV     #ERRME3,OUT.R0
 45 005631  021732                          CALL    TESTEW
```

```
GET SUBUNIT CHARACTERISTICS

                                                .SBTTL  GET SUBUNIT CHARACTERISTICS
                                        ;       GET SUBUNIT CHARACTERISTICS

005632  104301  002645                T4A:    MOV     UNITNB,R1               ; FIND SUBUNIT MASK FOR COMMAND
005634  103201  177774                        BIC     #^C3,R1                 ; CLEAR UNUSED BITS OF INDEX
005636  104207  000010                        MOV     #10,R0
005640  110207                        1$:     ROL     R0                      ; ROTATE
005641  117401                                DEC     R1                      ;  UNTIL DONE
005642  035640                                BPL     1$
005643  103207  177417                        BIC     #LBHINB,R0              ; CLEAR UNUSABLE BITS
005645  104070  002744                        MOV     R0,SUBUNT
005647  104203  002674                        MOV     #CR.SCR,R3              ; POINT TO GET SUBUNIT CHARACTERISTICS
005651  104200  002647  002647              MOV     #MS.SCR,COMND           ; SET UP FOR ERROR
005654  021076                              CALL    TALK                    ; SDI INTERCHANGE
005655  115002                              TST     R2                      ; SEE IF ERROR OCCURRED
005656  015660                              BEQ     2$
005657  021746                              CALL    TESTEX


                                                .SBTTL  SEEK TO CYLINDER 0, GROUP 0
                                        ;SEEK TO CYLINDER 0, GROUP 0

005660  104201  003076                2$:     MOV     #TSTCYL,R1
005662  114000  003076                        CLR     TSTCYL                  ; LO ORDER CYL 0
005664  104300  003007  003077              MOV     SUB+LBNCYL+1,TSTCYL+1   ; HI ORDER CYL 0
005667  103200  007777  003077              BIC     #HBHINB,TSTCYL+1
005672  114000  003100                        CLR     TSTCYL+2                ; GROUP 0
005674  022156                              CALL SEEK
```

```
                                        .SBTTL  CALCULATE NUMBERS
                                ;       CALCULATE NUMBER OF SECTORS PER TRACK
                                ;       NO. OF SECTORS/TRACK =(NO. OF LBN'S/TRACK) + (NO. OF RBN'S/TRACK)

005675  104301  003012        T4A1:   MOV SUB+RBNTRK,R1               ; GET RBN'S/TRACK
005677  103201  177600                BIC #177600,R1
005701  104307  003017                MOV SUB+LBNTRK,R0               ; GET LBN'S/TRACK
005703  103207  177400                BIC #HIBYTE,R0                  ; ZERO UPPER BITS
005705  105071                        ADD R0,R1                       ; ADD NUMBER OF LBN'S TO RBN'S
005706  104010  003073                MOV R1,SECTRK                   ; SAVE
005710  105011                        ADD R1,R1                       ; COMPUTE SECTORS TIMES TWO
005711  104010  003033                MOV R1,NSCSL                    ; AS SECTORS READ BEFORE DECLARING
                                                                        ; NO HEADER FOUND

                                ;COMPUTE FIRST CYLINDER IN XBN AREA

005713  104301  003006                MOV     SUB+LBNCYL,R1
005715  104010  003071                MOV     R1,FXBNCYL              ; FIRST XBN CYLINDER (LO)
005717  104307  003007                MOV     SUB+LBNCYL+1,R0
005721  103207  170000                BIC     #^CHBHINB,R0            ;CLEAR SUBUNIT CYL BITS
005723  104070  003072                MOV     R0,FXBNCYL+1            ; FIRST XBN CYLINDER (HI)

                                ;COMPUTE FIRST CYLINDER IN DBN AREA

005725  105301  003027                ADD SUB+XBNCYL,R1               ; ADD NUMBER OF XBN CYLINDERS
005727  045731                        BCC T4B                         ; BRANCH IF NO CARRY
005730  115407                        INC R0                          ; INCREMENT HI ORDER CYL
005731  104010  003067        T4B:    MOV R1,FDIACYL                  ; STORE STARTING DBN CYLINDER
005733  104070  003070                MOV R0,FDIACYL+1

                                ;COMPUTE LAST CYLINDER IN DBN AREA

005735  104303  003030                MOV     SUB+DBNCYL,R3           ; GET NUMBER OF DBN CYLINDERS
005737  110703                        SWAB    R3
005740  103203  177400                BIC     #HIBYTE,R3
005742  055764                        BNE     T4C
005743                                HARDER  17                      ;CHARACTERISTICS SAY LESS THAN 1 DIAGNOSTIC CYLINDER
005743  104200  000677  001037                        MOV     #MS17,OUT.04
005746  104300  001475  001036                        MOV     LUNIT,OUT.03
005751  104202  105711                                MOV     #17!ERHARD+3000.,R2
005753  104020  001035                                MOV     R2,OUT.02
005755  104200  005755  001034                        MOV     #.,OUT.01
005760  104200  060013  001033                        MOV     #ERRMES,OUT.RQ
005763  021740                        CALL    TESTEV
005764                          T4C:
                                ; NO LONGER NEED LAST DIAG CYL - JUST SAVE STARTING DBN, LAST TRACK, LAST GROUP, LAST CYL
                                ;               AND USE COMPUT XFC HEAVILY
005764  117403                        DEC     R3                      ; REDUCE BY ONE AND ADD TO
005765  105031                        ADD     R3,R1                   ; FIRST DBN CYLINDER TO
005766  045770                        BCC     T4D                     ; COMPUTE LAST CYLINDER
005767  115407                        INC     R0
005770  104010  003064        T4D:    MOV     R1,LDIACYL              ; STORE LAST DBN CYLINDER
005772  104070  003065                MOV     R0,LDIACYL+1
005774  104303  003010                MOV     SUB+GRPCYL,R3           ; GET GROUPS/CYLINDER
005776  103203  177400                BIC     #HIBYTE,R3              ; CLEAR UNUSED BITS
006000  117403                        DEC     R3                      ; LAST GROUP
006001  104030  003066                MOV     R3,LDIACYL+2
```

```
                                        .SBTTL  READ ALL FACTORY FORMATTED TRACKS
                                ;READ ALL FACTORY FORMATTED TRACKS.
                                ;REPORT ERROR IF ALL SECTORS ON A TRACK READ WITH AN ERROR.
                                
                                ;COMPUTE DBN OF FIRST BLOCK ON FACTORY FORMATTED CYLINDER
                                
 006003  114005                 T6:     CLR     R5
 006004  104303  003011                 MOV     SUB+TRKGRP,R3           ; TRACKS/GROUP
 006006  103203  177400                 BIC     #HIBYTE,R3
 006010  104304  003073                 MOV     SECTRK,R4               ; # SECTORS/TRACK
 006012  105035                 T6A:    ADD     R3,R5                   ; TO COMPUTE NUMBER OF SECTORS PER GROUP
 006013  117404                         DEC     R4
 006014  056012                         BNE     T6A
 006015  104050  003106                 MOV     R5,SECGRP               ; SAVE SECTORS/GROUP
 006017                                 PUSH    R2                      ; SAVE R2
 006017  100462                                                                 MOV R2,-(SP)
 006020  104302  003010                 MOV     SUB+GRPCYL,R2           ; GET GROUPS/CYL
 006022  103202  177400                 BIC     #HIBYTE,R2              ; CLEAR UNUSED BITS
 006024                                 PUSH    R2                      ; SAVE R2
 006024  100462                                                                 MOV R2,-(SP)
 006025  114000  003107                 CLR     SECCYL
 006027  105300  003106  003107 2$:     ADD     SECGRP,SECCYL           ; GET SECTORS/CYL
 006032  117402                         DEC     R2
 006033  056027                         BNE     2$
 006034                                 POP     R2                      ; RESTORE R2
 006034  104262                                                                MOV (SP)+,R2
 006035  104301  003030                 MOV     SUB+DBNCYL,R1           ; GET CYLINDERS IN DBN AREA
 006037  110701                         SWAB    R1
 006040  103201  177400                 BIC     #HIBYTE,R1              ; TO COMPUTE BLOCK NUMBER OF FIRST BLOCK
 006042  114005                         CLR     R5                      ; CLEAR PRODUCT
 006043  105025                 1$:     ADD     R2,R5
 006044  117401                         DEC     R1
 006045  056043                         BNE     1$
 006046                                 POP     R2                      ; RESTORE R2
 006046  104262                                                                MOV (SP)+,R2
 006047  114007                         CLR     R0
 006050                                 PUSH    R5                      ; SAVE NUMBER OF GROUPS IN DIAGNOSTIC AREA
 006050  100465                                                                MOV R5,-(SP)
 006051  117405                         DEC     R5                      ; SO DBN COMPUTED IS FIRST ON LAST GROUP
 006052  105301  003106         T6C:    ADD     SECGRP,R1
 006054  046056                         BCC     T6D
 006055  115407                         INC     R0
 006056  117405                 T6D:    DEC     R5
 006057  056052                         BNE     T6C
 006060  104010  003062                 MOV     R1,LCDBN
 006062  104070  003063                 MOV     R0,LCDBN+1
 006064                                 POP     R5                      ; R5 IS NUMBER OF GROUPS IN DIAG AREA
 006064  104265                                                                MOV (SP)+,R5
 006065  104307  003030                 MOV     SUB+DBNCYL,R0           ; GET NUMBER OF READ ONLY GROUPS
 006067  103207  177400                 BIC     #HIBYTE,R0              ; CLEAR UNUSED BITS
 006071  056113                         BNE     1$                      ; IF READ ONLY GROUPS > 0 THEN BRANCH
 006072                                 HARDER  14                      ; ZERO READ ONLY GROUPS
 006072  104200  000501  001037                 MOV     #MS14,OUT.04
 006075  104300  001475  001036                 MOV     LUNIT,OUT.03
 006100  104202  105706                         MOV     #14!ERHARD+3000.,R2
 006102  104020  001035                         MOV     R2,OUT.02
 006104  104200  006104  001034                 MOV     #.,OUT.01
```

```
                006107  104200  060013  001033
                                                                        MOV     #ERRMES,OUT.RQ
     47         006112  001740                          BR      TESTEV
     48         006113  104300  003064  003057  1$:     MOV     LDIACYL,ROFDC
     49         006116  104300  003065  003060          MOV     LDIACYL+1,ROFDC+1
     50         006121  107075                          SUB     R0,R5           ; R5 HAS NUMBER OF READ/WRITE GROUPS
     51         006122  016124                          BEQ     2$              ; IF R/W GROUPS ZERO, ERROR
     52         006123  036145                          BPL     3$              ; IF GREATER THAN ZERO, BRANCH
     53         006124                          2$:     HARDER  15              ; NO READ/WRITE GROUPS
                006124  104200  000555  001037                                      MOV     #MS15,OUT.04
                006127  104300  001475  001036                                      MOV     LUNIT,OUT.03
                006132  104202  105707                                            MOV     #15!ERHARD+3000.,R2
                006134  104020  001035                                            MOV     R2,OUT.02
                006136  104200  006136  001034                                    MOV     #.,OUT.01
                006141  104200  060013  001033                                    MOV     #ERRMES,OUT.RQ
     54         006144  001740                          BR      TESTEV
     55         006145  104304  003010          3$:     MOV     SUB+GRPCYL,R4   ; GET NUMBER OF GROUPS/CYL
     56         006147  103204  177400                  BIC     #HIBYTE,R4      ; CLEAR UNUSED BITS
     57         006151  107047                  4$:     SUB     R4,R0           ; SUBTRACT NUMBER OF GROUPS/CYL
     58         006152  076163                          BMI     5$
     59         006153  016170                          BEQ     6$
     60         006154  107200  000001  003057          SUB     #1,ROFDC        ; DECREMENT STARTING READ ONLY CYL BY 1
     61         006157  046151                          BCC     4$              ; IF NO CARRY, LOOP
     62         006160  117400  003060                  DEC     ROFDC+1         ; PROPOGATE BORROW
     63         006162  006151                          BR      4$              ; BRANCH
     64         006163  104203  177777          5$:     MOV     #-1,R3          ; SET UP FOR COMPLEMENT
     65         006165  103073                          BIC     R0,R3           ; COMPLEMENT (1'S)
     66         006166  115403                          INC     R3              ; 2'S COMPLEMENT
     67         006167  104037                          MOV     R3,R0           ; RESTORE TO R0
     68         006170  104070  003061          6$:     MOV     R0,ROFDC+2      ; SAVE IN STARTING GROUP
     69         006172  114003                          CLR     R3              ; CLEAR LO ORDER FIRST READ ONLY DBN
     70         006173  114004                          CLR     R4              ; CLEAR HI ORDER FIRST READ ONLY DBN
     71         006174  105303  003106          7$:     ADD     SECGRP,R3       ; ADD SEC/GRP TO LO ORDER STRT SEC
     72         006176  046200                          BCC     8$              ; IF NO CARRY, BRANCH
     73         006177  115404                          INC     R4              ; PROPOGATE CARRY
     74         006200  117405                  8$:     DEC     R5              ; DECREMENT COUNT
     75         006201  056174                          BNE     7$              ; LOOP
     76         006202  104030  003055                  MOV     R3,ROFDBN
     77         006204  104040  003056                  MOV     R4,ROFDBN+1
     78         006206  104030  003104                  MOV     R3,CURBLK
     79         006210  104040  003105                  MOV     R4,CURBLK+1
     80         006212  104200  000104  003115          MOV     #'D,LETTER              ; MAKE DBN'S
     81                                          ; POINT TO CURBLK, CALL SOME SUB THAT LOOKS AT THE POINTER PASSED
     82                                          ; AND THE 'LETTER' AND COMPUTS CYL, TRK, AND GRP. THEN RETURN
     83                                          ; THEN SEE IF CYL COMPUTED IS > MAX CYL IF SO, THIS PHASE OF TEST IS OVER (BR TGX)
     84         006215  104205  003104          T6G:    MOV     #CURBLK,R5      ;
     85         006217  021776                          CALL    FNDCY2          ; GO DIRECTLY TO DBN WILL BE PROCESSED
     86                                          ; THEN SEE IF CYL COMPUTED IS > MAX CYL. IF SO, THIS PHASE OF TEST IS OVER (BR TGX)
     87         006220  106300  003065  003077  1$:     CMP     LDIACYL+1,TSTCYL+1
     88         006223                                  BCS     T6F
                006223  046225                                                                    BCC     64$
                006224  006255                                                                    BR      T6F
     89         006225  106300  003064  003076          CMP     LDIACYL,TSTCYL
     90         006230                                  BCS     T6F
                006230  046232                                                                    BCC     65$
                006231  006255                                                                    BR      T6F
     91         006232  104201  003076                  MOV     #TSTCYL,R1
     92         006234  022156                          CALL    SEEK            ; SEEK TO REQUESTED CYL
```

```
     93 006235  104205  003104                  MOV     #CURBLK,R5
     94 006237  104303  003073                  MOV     SECTRK,R3
     95 006241  022331                  T6E:    CALL    READ1                   ;READ EACH SECTOR TILL ONE READS OK
     96 006242  103200  177400  003105          BIC     #HIBYTE,CURBLK+1
     97 006245  105300  003073  003104          ADD     SECTRK,CURBLK
     98 006250  046215                          BCC     T6G
     99 006251  115400  003105                  INC     CURBLK+1
    100 006253  115404                          INC     R4
    101 006254  006215                          BR      T6G
    102
    103 006255                          T6F:
```

```
                                        .SBTTL  SEEK TO SELECTED GROUP, READ HDR, SEEK TO DIAGNOSTIC AREA
                                ;STARTING WITH CYLINDER 0, GROUP 0 AND INCREMENTING THROUGH EVERY GROUP
                                ;ON THE DISK, PERFORM A SEEK TO THE SELECTED GROUP, READ A HEADER
                                ;ON THAT GROUP AND THEN A SEEK TO THE FACTORY FORMATTED DIAGNOSTIC
                                ;GROUP TO VERIFY HEADS POSITIONED CORRECTLY.

                                ;COMPUTE LBNS PER GROUP

006255  114001                  T7:     CLR     R1                      ; LBN'S/TRK X TRK/GROUP
006256  104305  003011                  MOV     SUB+TRKGRP,R5
006260  103205  177400                  BIC     #HIBYTE,R5
006262  104303  003017                  MOV     SUB+LBNTRK,R3           ;GET LBN'S PER TRACK
006264  103203  177400                  BIC     #HIBYTE,R3              ;CLEAR UNUSED BITS
006266  104030  003101                  MOV     R3,BLOCKT
006270  105031                  T7A:    ADD     R3,R1
006271  117405                          DEC     R5
006272  056270                          BNE     T7A
006273  104010  003102                  MOV     R1,BLOCKG               ;SAVE IN BLOCKG
006275  114000  003103                  CLR     BLOCKC
006277  104307  003010                  MOV     SUB+GRPCYL,R0
006301  103207  177400                  BIC     #HIBYTE,R0
006303  105300  003102  003103  1$:     ADD     BLOCKG,BLOCKC
006306  117407                          DEC     R0
006307  056303                          BNE     1$
006310  114000  003076                  CLR     TSTCYL
006312  114000  003100                  CLR     TSTCYL+2                ;GROUP ZERO
006314  114000  003074                  CLR     TSTBLK
006316  114000  003075                  CLR     TSTBLK+1
006320  114000  003121                  CLR     GRPCNT
006322  114000  003122                  CLR     OLDGRP                  ; CLEAR OLD GROUP VALUE USED IN FNDCYL
006324  104200  000114  003114          MOV     #'L,AREA                ;AREA BEING SEEKED INTO IS LBNS
006327  104300  003114  003115  T7C:    MOV     AREA,LETTER             ; MOVE TO LETTER FOR REPORTING
006332  104205  003074                  MOV     #TSTBLK,R5              ;POINT TO LBN NUMBER FOR -> VAR1
                                ; COMPUTE CYL, TRK AND GRP.
                                ; IF CYL < STARTING CYL THEN CONTINUE ELSE IF CYL >= STARTING DIAG CYL THEN OUT (BR T7X)
                                ; ELSE IF CYL = STARTING XBN CYL CLR TSTBLK, TSTBLK+1 AND BR T7C
006334  021770                          CALL    FNDCYL                  ;FIND CYLINDER
006335  106300  003071  003131          CMP     FXBNCYL,CYLLO           ;IS CYL = STARTING XBN?
006340  056374                          BNE     1$                      ;IF NOT, BRANCH
006341  106300  003072  003132          CMP     FXBNCYL+1,CYLLO+1       ;
006344  056374                          BNE     1$
006345  106200  000130  003114          CMP     #'X,AREA                ; IF HERE AGAIN, BRANCH
006350  016374                          BEQ     1$
006351  104200  000130  003114          MOV     #'X,AREA                ; NOW IN XBN AREA
006354  114000  003074                  CLR     TSTBLK                  ;IF SO, CLEAR TSTBLK
006356  114000  003075                  CLR     TSTBLK+1
006360  104300  003073  003101          MOV     SECTRK,BLOCKT           ;CHANGE BLOCK COUNT PER TRACK
006363  104300  003106  003102          MOV     SECGRP,BLOCKG           ;CHANGE BLOCK COUNT PER GROUP
006366  104300  003107  003103          MOV     SECCYL,BLOCKC           ;CHANGE BLOCK COUNT PER CYL
006371  114000  003122                  CLR     OLDGRP                  ; CLEAR OLD GROUP
006373  006327                          BR      T7C                     ; AND TRY AGAIN WITH XBN
006374  106300  003067  003076  1$:     CMP     FDIACYL,TSTCYL          ;DONE?
006377  056404                          BNE     2$                      ;IF NOT, CONTINUE
006400  106300  003070  003077          CMP     FDIACYL+1,TSTCYL+1      ;DONE?
006403  016476                          BEQ     T7X                     ;IF SO, EXIT
006404  104300  003114  003115  2$:     MOV     AREA,LETTER
006407  104201  003076                  MOV     #TSTCYL,R1              ;SEEK TO CYLINDER
```

```
     58 006411  022156                          CALL    SEEK
     59 006412  104205  003074                  MOV     #TSTBLK,R5              ; SET UP POINTER TO TEST BLOCK
     60 006414  104303  003073                  MOV     SECTRK,R3               ;GET MAXIMUM NUMBER OF SECTORS TO READ
     61 006416  022331                          CALL    READ1                   ;READ UNTIL AT LEAST ONE RECORD READ WITHOUT ERROR
     62                                                         COMPUTE CYL FROM LCDBN
     63 006417  104200  000104  003115  3$:     MOV     #'D,LETTER              ; IN DIAGNOSTIC AREA
     64 006422  021770                          CALL    FNDCYL
     65 006423  104201  003076                  MOV     #TSTCYL,R1
     66 006425  104303  003073                  MOV     SECTRK,R3               ; R3 = SECTORS/TRACK
     67 006427  022156                          CALL    SEEK
     68 006430  104205  003062                  MOV     #LCDBN,R5               ; SET UP POINTER TO TEST BLOCK
     69 006432  022331                          CALL    READ1                   ;READ UNTIL AT LEAST ONE RECORD READ WITHOUT ERROR
     70 006433  115400  003121                  INC     GRPCNT
     71 006435  103200  177400  003121          BIC     #HIBYTE,GRPCNT
     72 006440  056461                          BNE     5$
     73 006441                                  PUSH    R0                      ; SAVE R0
        006441  100467                                                                     MOV R0,-(SP)
     74 006442  104207  060007                  MOV     #T4SOFT,R0              ; TO INTERRUPT HOST <<ONLY>>
     75 006444  104300  001475  001034          MOV     LUNIT,OUT.01            ; IDENTIFY UNIT
     76 006447  114000  001035                  CLR     OUT.02                  ; NO SOFT ERRORS
     77 006451  114000  001036                  CLR     OUT.03                  ; NO ECC CORRECTIONS
     78 006453  114000  001037                  CLR     OUT.04
     79 006455  114000  001040                  CLR     OUT.05
     80 006457  021000                          CALL    HOSTRQ                  ; REPORT (NOTHING)
     81 006460                                  POP     R0                      ; RESTORE R0
        006460  104267                                                                     MOV (SP)+,R0
     82 006461  105300  003103  003074  5$:     ADD     BLOCKC,TSTBLK
     83 006464  046467                          BCC     6$
     84 006465  115400  003075                  INC     TSTBLK+1
     85 006467  105300  003102  003074  6$:     ADD     BLOCKG,TSTBLK
     86 006472  046475                          BCC     7$
     87 006473  115400  003075                  INC     TSTBLK+1
     88 006475  006327                  7$:     BR      T7C
```

```
                                T7X:
                                        BR      OVRLAY          ; BRING IN REST OF THE CODE

                        006475  OVL.SU  =       .-TEST
```

```
                                .SBTTL  *****DM OVERLAY 2*****CHECK WRITE PROTECT
                                DMOVLY  TS,OVRLPT
                                .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                        ;***************************************************************************
                        ;***************************************************************************
                        ;***************************************************************************
                        ;***************************************************************************
000002                  TEST    =       2
```

```
     5 005262  104200  177777  003117          MOV     #-1,WFLAG               ; SET FLAG
     6 005265  114000  003104                  CLR     CURBLK                  ; CLEAR BLOCK
     7 005267  114000  003105                  CLR     CURBLK+1                ;
     8 005271  114000  003111                  CLR     CURTRK                  ; CLEAR TRACK
     9 005273  104300  003067  003076          MOV     FDIACYL,TSTCYL          ; POINT TO CYL TO TEST
    10 005276  104300  003070  003077          MOV     FDIACYL+1,TSTCYL+1
    11 005301  114000  003100                  CLR     TSTCYL+2                ; CLEAR GROUP
    12 005303  104201  003076                  MOV     #TSTCYL,R1              ; R1 -> TEST CYL
    13 005305  022156                          CALL    SEEK                    ; SEEK THERE
    14 005306  027020                          CALL    WRITEB                  ; AND ATTEMPT TO WRITE
    15 005307  115007                          TST     R0                      ; WAS IT SUCCESSFUL?
    16 005310  055334                          BNE     1$                      ; IF NOT, THAT WAS EXPECTED
    17 005311  115001                          TST     R1                      ;
    18 005312  055334                          BNE     1$                      ;
    19 005313                                  HARDER  35                      ; REPORT ERROR
       005313  104200  002245  001037                          MOV     #MS35,OUT.04
       005316  104300  001475  001036                          MOV     LUNIT,OUT.03
       005321  104202  105733                                  MOV     #35!ERHARD+3000.,R2
       005323  104020  001035                                  MOV     R2,OUT.02
       005325  104200  005325  001034                          MOV     #.,OUT.01
       005330  104200  060013  001033                          MOV     #ERRMES,OUT.RQ
    20 005333  021740                          CALL    TESTEV
    21 005334  021761                  1$:     CALL    DR.CLR                  ; CLEAR DRIVE
    22 005335  114000  003117                  CLR     WFLAG                   ; CLEAR FLAG
```

```
                                                .SBTTL  CHANGE MODE TO ALLOW FORMATTING AND WRITING
005337                                  T8:
                                        ;
                                        ;       FIRST CHANGE MODE TO ALLOW FORMATTING AND WRITING
                                        ;
005337  104204  000004                  MOV     #4,R4                   ; R4 HAS DECREMENT COUNTER
005341  104203  002706                  2$:     MOV     #CR.MOD,R3              ; POINT TO CHANGE MODE COMMAND
005343  104300  002757  002755                  MOV     MODE2,MODE              ; WRITE PROTECT, DIAG CYL ACCESS, NO FORMATTING
005346  104200  002676  002647                  MOV     #MS.MOD,COMND           ; SET UP FOR ERROR
005351  021076                                  CALL    TALK                    ; INITIATE SDI INTERCHANGE
005352  115002                                  TST     R2                      ; SEE IF ERROR OCCURRED
005353  015360                                  BEQ     1$                      ; IF NOT, BRANCH
005354  021746                                  CALL    TESTEX                  ; IF SO, ERROR
005355  117404                                  DEC     R4                      ; DONE?
005356  055341                                  BNE     2$                      ; IF NOT, BRANCH (TRY AGAIN BECAUSE FORMAT
                                                                                ;       WON'T WORK WITH A WRITE PROTECTED DRIVE)
005357  001617                                  BR      TESTX                   ; ELSE, EXIT
005360  114000  003120                  1$:     CLR     FLAG                    ; USE FLAG TO DECIDE IF WE FORMAT OR TEST A TRACK
005362                                  T08:
                                                .SBTTL  FORMAT A TRACK THEN CHECK IT.
                                        ;
                                        ;       FORMAT ALL TRACKS, THEN DO SEVERAL WRITES, READS AND DATA COMPARES
                                        ;       UNTIL ONE SECTOR PASSES.  DO THIS FOR EACH TRACK IN EACH GROUP
                                        ;       ON EACH CYLINDER IN THE DIAGNOSTIC AREA EXCEPT FOR THE LAST
                                        ;       (FACTORY FORMATTED) CYLINDER ON THE PACK
                                        ;
005362  104300  003067  003076                  MOV     FDIACYL,TSTCYL          ; START AT FIRST DIAGNOSTIC CYL
005365  104300  003070  003077                  MOV     FDIACYL+1,TSTCYL+1
005370  104201  003076                          MOV     #TSTCYL,R1              ; POINT TO CYLINDER TO SEEK TO
005372  114000  003100                          CLR     TSTCYL+2                ; START WITH GROUP 0
005374  022156                                  CALL    SEEK                    ; SEEK TO FIRST DIAGNOSTIC CYLINDER
005375  114004                                  CLR     R4                      ; START WITH DBN 0
005376  104040  003074                          MOV     R4,TSTBLK
005400  104040  003075                          MOV     R4,TSTBLK+1             ; CLEAR BLOCK
005402  104040  003111                  T8A:    MOV     R4,CURTRK               ; SAVE TRACK NUMBER
005404  115000  003120                          TST     FLAG                    ; DO WE FORMAT
005406  055411                                  BNE     1$                      ; IF NOT ZERO, TEST TRACK
005407  026465                                  CALL    FORTRK                  ; FORMAT THE TRACK
005410  005412                                  BR      2$                      ; DO NOT TEST TRACK UNTIL ALL FORMATTED
005411  026710                          1$:     CALL    TRKTST                  ; TEST THE TRACK
005412  105300  003073  003074          2$:     ADD     SECTRK,TSTBLK           ; ADD SECTORS/TRACK
005415  045420                                  BCC     T8B                     ; IF NO CARRY, BRANCH
005416  115400  003075                          INC     TSTBLK+1                ; INCREMENT
005420  104304  003111                  T8B:    MOV     CURTRK,R4               ; GET TRACK NUMBER
005422  115404                                  INC     R4                      ; INCREMENT TRACK NUMBER
005423  104303  003011                          MOV     SUB+TRKGRP,R3           ; GET NUMBER OF TRACKS/GROUP
005425  103203  177400                          BIC     #HIBYTE,R3              ; CLEAR UNUSED BITS
005427  106034                                  CMP     R3,R4                   ; SEE IF ALL TRACKS READ
005430  055402                                  BNE     T8A                     ; IF NOT, BRANCH
005431  114004                                  CLR     R4                      ; ZERO R4
005432  115400  003100                          INC     TSTCYL+2                ; MOVE TO NEXT GROUP
005434  104301  003010                          MOV     SUB+GRPCYL,R1           ; GET GROUPS/CYLINDER
005436  103201  177400                          BIC     #HIBYTE,R1              ; CLEAR UNUSED BITS
005440  106010  003100                          CMP     R1,TSTCYL+2             ; COMPARE
005442  055452                                  BNE     T8D                     ; IF ALL GROUPS NOT TESTED, BRANCH
005443  104040  003100                          MOV     R4,TSTCYL+2             ; START WITH GROUP 0 ON NEW CYLINDER
005445  115400  003076                          INC     TSTCYL                  ; INCREMENT CYLINDER
```

```
     58 005447  045452                          BCC     T8D                     ; IF NO CARRY, BRANCH
     59 005450  115400  003077                   INC     TSTCYL+1                ; INCREMENT HI CYLINDER
     60 005452  106300  003076  003057  T8D:     CMP     TSTCYL,ROFDC            ; SEE IF ON LAST CYLINDER (LO ORDER)
     61 005455  055466                          BNE     1$                      ; IF NOT, BRANCH
     62 005456  106300  003077  003060          CMP     TSTCYL+1,ROFDC+1        ; SEE IF ON LAST CYLINDER (HI ORDER)
     63 005461  055466                          BNE     1$                      ; IF NOT, BRANCH
     64 005462  106300  003100  003061          CMP     TSTCYL+2,ROFDC+2        ; SEE IF ON RESERVED GROUPS
     65 005465  015472                          BEQ     T8T                     ; IF SO, END THIS PHASE OF TEST
     66 005466  104201  003076          1$:     MOV     #TSTCYL,R1              ; POINT TO NEW CYLINDER OR GROUP
     67 005470  022156                          CALL    SEEK                    ; ISSUE SEEK
     68 005471  005402                          BR      T8A                     ; BRANCH
     69 005472  115000  003120          T8T:    TST     FLAG                    ; ARE WE DONE?
     70 005474  055501                          BNE     T8E                     ; IF SO, BRANCH
     71 005475  104200  177777  003120          MOV     #-1,FLAG                ; ELSE, CHANGE FLAG
     72 005500  005362                          BR      T08                     ; AND CHECK TRACKS
     73
     74                                  ; *** NOW SEND INVALID SELECT GROUP COMMAND AND EXPECT ERROR
     75 005501  104200  000001  001462  T8E:    MOV     #1,SDILTO               ; CHANGE LONG TIMEOUT
     76 005504  104207  003010                  MOV     #SUB+GRPCYL,R0          ; R0 HAS GROUP #
     77 005506  103207  177400                  BIC     #HIBYTE,R0              ; STRIP OFF UNUSED BITS
     78 005510  106207  000377                  CMP     #LOBYTE,R0              ; IS THIS UNIT USE MAX BITS?
     79 005512  015563                          BEQ     T11                     ; IF SO, BRANCH (DON'T DO TEST)
     80 005513  101207  107000                  BIS     #SL.GRP,R0              ; SET SELECT GROUP CODE
     81 005515  104070  002746                  MOV     R0,RUN                  ; STORE IN RUN CODE
     82 005517  114000  002726                  CLR     CR.RUN+1                ; SET UP FOR LEVEL 1 EXCHANGE
     83 005521  104203  002725                  MOV     #CR.RUN,R3              ; R3 -> PACKET
     84 005523  104302  002646                  MOV     SDI,R2                  ; SEND MESSAGE
     85 005525  104237                          MOV     (R3)+,R0                ; SET UP PARAMETERS
     86 005526  104131                          MOV     (R3),R1                 ;
     87 005527  060004                          XFC     SEND                    ; SEND COMMAND
     88 005530  115001                          TST     R1                      ; IF FAIL, BRANCH
     89 005531  055562                          BNE     T8F
     90 005532  104203  002701                  MOV     #CR.GST,R3              ; CHECK OUT STATUS
     91 005534  021076                          CALL    TALK
     92 005535  102200  000350  002766          BIT     #<ST.PE+ST.RE+ST.FE+ST.WE>,ST.ERR+ST ;ERROR?
     93 005540  055562                          BNE     T8F                     ; IF SO, BRANCH
     94 005541                                  HARDER  33                      ; ELSE ERROR
        005541  104200  002156  001037                                                  MOV     #MS33,OUT.04
        005544  104300  001475  001036                                                  MOV     LUNIT,OUT.03
        005547  104202  105731                                                          MOV     #33!ERHARD+3000.,R2
        005551  104020  001035                                                          MOV     R2,OUT.02
        005553  104200  005553  001034                                                  MOV     #.,OUT.01
        005556  104200  060013  001033                                                  MOV     #ERRMES,OUT.RQ
     95 005561  021740                          CALL    TESTEV
     96 005562  021761                  T8F:    CALL    DR.CLR                  ; CLEAR DRIVE OF ERROR
```

```
                                        .SBTTL  SEND INVALID LEVEL 2 COMMANDS
005563  114000  002647          T11:    CLR     COMND                   ; CLEAR COMMAND CODE AS A FLAG
005565  114007                          CLR     R0                      ; CLEAR SDI COMMAND VALUE
005566  104200  000001  002726          MOV     #1,CR.RUN+1             ; RESET PROPER LENGTH
005571  104070  002746          1$:     MOV     R0,RUN                  ; SET CODE
005573                                  PUSH    R0                      ; SAVE R0
005573  100467                                                                  MOV R0,-(SP)
005574  026343                          CALL    TSTCMD                  ; AND TEST
005575                                  POP     R0                      ; RESTORE R0
005575  104267                                                                  MOV (SP)+,R0
005576  115007                          TST     R0                      ; TRY INVALID OPCODE ONCE
005577  055603                          BNE     2$                      ; EXIT WHEN TRIED ONCE WITH INVALID CODE
005600  105207  000400                  ADD     #400,R0                 ; ADD TO NEXT CODE
005602  055571                          BNE     1$                      ; IF NOT DONE, TEST AGAIN
005603  104200  000377  002746  2$:     MOV     #377,RUN                ; TRY WITH LOW BITS SET
005606  026343                          CALL    TSTCMD                  ;
005607  104300  002760  002746          MOV     RUNCMD,RUN              ; RESTORE RUN COMMAND
005612  104205  000100                  MOV     #64.,R5
005614  104050  002726                  MOV     R5,CR.RUN+1             ; STORE INVALID COMMAND LENGTH
005616  104204  003415                  MOV     #SER50,R4               ;
005620  026343                          CALL    TSTCMD
005621  114005                          CLR     R5                      ; TRY AGAIN WITH 0 COMMAND LENGTH BUT INVALID COMMAN
005622  104204  003422                  MOV     #SER51,R4
005624  104050  002704                  MOV     R5,CR.GST+3             ; DESTROY RESPONSE LENGTH
005626  104203  002701                  MOV     #CR.GST,R3
005630  026345                          CALL    TSTCM2
005631  104200  000007  002704          MOV     #7,CR.GST+3             ; RESTORE COMMAND LENGTH
```

```
                                        .SBTTL  SEND INVALID LEVEL 1 COMMANDS

                                ; *** SEND START THEN SELECT GROUP
005634  114000  002726                  CLR     CR.RUN+1
005636  104200  070400  002746          MOV     #MS.STR,RUN             ; SET UP COMMAND
005641  026330                          CALL    SNDLV1                  ; SEND LEVEL 1 START COMMAND
005642  104200  003430  002647          MOV     #SER52,COMND
005645  104200  107000  002746          MOV     #SL.GRP,RUN             ; MOVE SELECT GROUP CODE INTO COMMAND
005650  026333                          CALL    SNDL1A                  ; SEND NEXT FRAME
005651  104200  131000  002746          MOV     #MS.END,RUN             ; SET UP END FRAME
005654  026343                          CALL    TSTCMD                  ; CHECK IF CAUGHT

                                ; *** SEND END WITH NO START
005655  104200  131000  002746          MOV     #MS.END,RUN             ; SET UP END FRAME
005660  104200  003467  002647          MOV     #SER53,COMND
005663  026343                          CALL    TSTCMD                  ; CHECK IF CAUGHT

                                ; *** SEND START THEN END WITH BAD CHECKSUM
005664  104200  070400  002746          MOV     #MS.STR,RUN             ; SET UP START FRAME
005667  026330                          CALL    SNDLV1                  ; SEND FRAME
005670  104200  003520  002647          MOV     #SER54,COMND
005673  104200  131000  002746          MOV     #MS.END,RUN             ; SET UP END FRAME (ZERO CHECKSUM)
005676  026343                          CALL    TSTCMD                  ; CHECK IF CAUGHT

                                ; *** SEND CONTINUE WITH NO START
005677  104200  152000  002746          MOV     #MS.CNT,RUN             ; SET CONTINUE FRAME
005702  101200  000014  002746          BIS     #RUNLBC,RUN
005705  026330                          CALL    SNDLV1                  ; SEND IT
005706  104200  131000  002746          MOV     #MS.END,RUN             ; SET UP END FRAME
005711  104200  003551  002647          MOV     #SER55,COMND
005714  026343                          CALL    TSTCMD                  ; CHECK IF CAUGHT
```

```
                                ;
                                ;       FIND IF DRIVE WAS FORMATTED IN 512 BYTE MODE.
                                ;
005715  104307  002761      T5:   MOV     COPY,R0                 ; GET NUMBER OF XBN COPIES
005717  105077                    ADD     R0,R0                   ; DOUBLE
005720  104070  002763            MOV     R0,SERRTY               ; RETRY 2 TIMES THE NUMBER OF COPIES
005722  114000  002762      1$:   CLR     RETRY                   ; TO SET UP COPIES AND SETUP CALC
005724  114000  003104            CLR     CURBLK                  ; PUT LO ORDER XBN IN CURBLK
005726  114000  003105            CLR     CURBLK+1                        ; PUT HI ORDER XBN IN CURBLK+1
005730  104205  003104      2$:   MOV     #CURBLK,R5
005732  104204  003067            MOV     #FDIACYL,R4
005734  104303  003073            MOV     SECTRK,R3
005736  114001                    CLR     R1
005737  021770                    CALL    FNDCYL
005740  104302  002646            MOV     SDI,R2
005742  104203  002720            MOV     #CR.INR,R3              ; SET UP FOR RECALIBRATE
005744  021076                    CALL    TALK                    ; RECALIBRATE DRIVE
005745  115002                    TST     R2                      ; SEE IF ERROR OCCURRED
005746  056077                    BNE     8$                      ; IF SO, BRANCH
005747  104201  003131            MOV     #CYLLO,R1
005751  022156                    CALL    SEEK
005752  104301  001462      3$:   MOV     SDILTO,R1               ; INNER LOOP TIMEOUT
005754  114005                    CLR     R5
005755  020746            4$:   CALL    RTDSL                   ; GET REAL TIME DRIVE STATE
005756  115002                    TST     R2                      ; SEE IF ERROR OCCURRED
005757  056077                    BNE     8$                      ; IF SO, BRANCH
005760  115001                    TST     R1                      ; SEE IF READ/WRITE READY IS ASSERTED
005761  075767                    BMI     5$                      ; IF SO, BRANCH
005762                            ASSUME  RWRDY,100000
005762  117405                    DEC     R5
005763  055755                    BNE     4$
005764  117404                    DEC     R4
005765  055755                    BNE     4$
005766  006077                    BR      8$                      ; ERROR IF TIMED OUT
005767  104200  100000  003137  5$:   MOV     #RSTOP,CHAIN+RW.STAT    ; MOVE LAST CHAIN FLAG TO CHAIN
005772  104300  003104  003141        MOV     CURBLK,CHAIN+RW.LOW     ; MOVE TO OUTPUT
005775  104301  003010                MOV     SUB+HIXBN,R1            ; GET HI XBN BITS
005777  110601                        ROR     R1                      ; ROTATE TO CORRECT POSITION
006000  110601                        ROR     R1                      ; ROTATE TO CORRECT POSITION
006001  110601                        ROR     R1                      ; ROTATE TO CORRECT POSITION
006002  110601                        ROR     R1                      ; ROTATE TO CORRECT POSITION
006003  103201  170377                BIC     #HBLONB,R1              ; CLEAR UNUSED BITS
006005  101201  120000                BIS     #HD.XBN,R1              ; MAKE A XBN
006007  105301  003105                ADD     CURBLK+1,R1             ; SET IN HI XBN BITS
006011  104010  003142                MOV     R1,CHAIN+RW.HI          ; SAVE
006013  104300  003134  003143        MOV     TRACK,CHAIN+RW.CMD      ; MOVE TRACK TO CHAIN
006016  101200  013400  003143        BIS     #RREAL,CHAIN+RW.CMD     ; SET IN REAL TIME COMMAND
006021  104207  003665                MOV     #RBUFD,R0
006023  104070  003140                MOV     R0,CHAIN+RW.BUF
006025  104302  002646                MOV     SDI,R2                  ; R2 HAS UDA PORT MASK
006027  104207  003137                MOV     #CHAIN,R0               ; R0 POINTS TO CHAIN
006031  060012                        XFC     WAITSI                  ; WAIT FOR SECTOR OR INDEX
006032  115001                        TST     R1                      ; SEE IF ERROR OCCURRED
006033  056077                        BNE     8$                      ; IF SO, BRANCH
006034  060002                        XFC     XREAD                   ; READ THE SECTOR
006035  106201  000004                CMP     #4,R1                   ; SEE IF XBN HEADER COMPARE FAILURE
006037  016062                        BEQ     7$                      ; IF SO, TRY NEXT COPY
```

```
     78 006040  115001                          TST     R1                      ; SEE IF AN ERROR OCCURRED
     79 006041  056077                          BNE     8$                      ; IF SO, BRANCH
     83 006042  102200  010000  003137          BIT     #ECCFLG,CHAIN+RW.STAT   ; SEE IF BUFFER HAS AN ECC ERROR
     92 006045  016061                          BEQ     6$                      ; IF NOT, BRANCH
     93 006046  104207  003137                  MOV     #CHAIN,R0               ; POINT TO CHAIN
     94 006050  060015                          XFC     ECC                     ; CORRECT THE BUFFER
     95 006051  115001                          TST     R1                      ; SEE IF ERROR
     96 006052  056062                          BNE     7$                      ; IF SO, BRANCH
     97 006053  104307  002766                  MOV     ST+ECCRSH,R0
     98 006055  103207  177400                  BIC     #HIBYTE,R0
     99 006057  106071                          CMP     R0,R1
    100 006060  076062                          BMI     7$
    101 006061  006126                  6$:     BR      9$                      ;
    102 006062  115400  002762          7$:     INC     RETRY                   ; INCREMENT CURRENT COPY NUMBER
    103 006064  106300  002761  002762          CMP     COPY,RETRY              ; SEE IF ALL COPIES TRIED
    104 006067  016077                          BEQ     8$                      ; IF SO, BRANCH
    105 006070  105300  003003  003104          ADD     CR+FCTSIZ,CURBLK        ; ADD TO CURRENT BLOCK
    106 006073  045730                          BCC     2$                      ; CALCULATE AND TRY NEXT SECTOR
    107 006074  115400  003105                  INC     CURBLK+1                        ; PROPOGATE CARRY
    108 006076  005730                          BR      2$                      ; BRANCH
    109 006077  117400  002763          8$:     DEC     SERRTY                  ; DECREMENT RETRY COUNT
    110 006101  055722                          BNE     1$                      ; IF UNEXHAUSTED, BRANCH
    111 006102                                  HARDER  38,#SER36
        006102  104200  002422  001037                                                  MOV     #MS38,OUT.04
        006105  104200  002446  001040                                                  MOV     #SER36,OUT.05
        006110  104300  001475  001036                                                  MOV     LUNIT,OUT.03
        006113  104202  105736                                                          MOV     #38!ERHARD+3000.,R2
        006115  104020  001035                                                          MOV     R2,OUT.02
        006117  104200  006117  001034                                                  MOV     #.,OUT.01
        006122  104200  060013  001033                                                  MOV     #ERRMES,OUT.RQ
    112 006125  006202                          BR      11$
    113 006126  106200  126736  003665  9$:     CMP     #MOD512,RBUFD           ; SEE IF 512 BYTE MODE DRIVE
    114 006131  016203                          BEQ     10$                     ; IF SO, BRANCH
    115 006132  106200  074161  003665          CMP     #MOD576,RBUFD           ; SEE IF 576 BYTE MODE DRIVE
    116 006135  016162                          BEQ     12$
    117 006136                                  HARDER  36,#SER36               ; SET UP REPORT
        006136  104200  002306  001037                                                  MOV     #MS36,OUT.04
        006141  104200  002446  001040                                                  MOV     #SER36,OUT.05
        006144  104300  001475  001036                                                  MOV     LUNIT,OUT.03
        006147  104202  105734                                                          MOV     #36!ERHARD+3000.,R2
        006151  104020  001035                                                          MOV     R2,OUT.02
        006153  104200  006153  001034                                                  MOV     #.,OUT.01
        006156  104200  060013  001033                                                  MOV     #ERRMES,OUT.RQ
    118 006161  006202                          BR      11$
    119 006162                          12$:    HARDER  37
        006162  104200  002331  001037                                                  MOV     #MS37,OUT.04
        006165  104300  001475  001036                                                  MOV     LUNIT,OUT.03
        006170  104202  105735                                                          MOV     #37!ERHARD+3000.,R2
        006172  104020  001035                                                          MOV     R2,OUT.02
        006174  104200  006174  001034                                                  MOV     #.,OUT.01
        006177  104200  060013  001033                                                  MOV     #ERRMES,OUT.RQ
    120 006202  021753                  11$:    CALL    TESTED
    121 006203                          10$:
```

```
                                                .SBTTL  ISSUE DISCONNECT COMMAND
006203  104203  002667          T9:     MOV     #CR.DIS,R3              ; POINT TO DISCONNECT COMMAND
006205  104200  002621  002647          MOV     #MS.DIS,COMND           ; SET UP FOR ERROR
006210  021076                          CALL    TALK                    ; INITIATE SDI INTERCHANGE
006211  115002                          TST     R2                      ; SEE IF ERROR OCCURRED
006212  016214                          BEQ     T10
006213  021746                          CALL    TESTEX                  ; IF SO, BRANCH

                                                .SBTTL  CHECK AVAIL FLAG
006214  114005          T10:    CLR     R5                      ; SET UP TIMEOUT COUNTER
006215  104302  002646  T10LOP: MOV     SDI,R2                  ; MOVE MASK TO R2
006217  020754                  CALL    RDSTAT                  ; GET DRIVE STATUS
006220  115002                  TST     R2                      ; WAS IT OK?
006221  016267                  BEQ     2$                      ; IF NO ERROR, BRANCH
006222  102201  000400          BIT     #RCVERR,R1              ; SEE IF ANY ERRORS
006224  056246                  BNE     1$                      ; IF NOT, ERROR
006225                          HARDER  11                      ; REPORT INVALID STATUS ERROR
006225  104200  000366  001037                  MOV     #MS11,OUT.04
006230  104300  001475  001036                  MOV     LUNIT,OUT.03
006233  104202  105703                          MOV     #11!ERHARD+3000.,R2
006235  104020  001035                          MOV     R2,OUT.02
006237  104200  006237  001034                  MOV     #.,OUT.01
006242  104200  060013  001033                  MOV     #ERRMES,OUT.RQ
006245  001732                  BR      TESTEW                  ; BRANCH TO DONE
006246                  1$:
006246                          HARDER  12                      ; REPORT XMIT ERROR
006246  104200  000434  001037                  MOV     #MS12,OUT.04
006251  104300  001475  001036                  MOV     LUNIT,OUT.03
006254  104202  105704                          MOV     #12!ERHARD+3000.,R2
006256  104020  001035                          MOV     R2,OUT.02
006260  104200  006260  001034                  MOV     #.,OUT.01
006263  104200  060013  001033                  MOV     #ERRMES,OUT.RQ
006266  001732                  BR      TESTEW                  ; BRANCH TO DONE
006267                  2$:
006267  102201  000100          BIT     #AVAIL,R1               ; SEE IF AVAILABLE IS ASSERTED
006271  056321                  BNE     T12                     ; IF SO, BRANCH TO LAST TEST
006272  117405                  DEC     R5                      ; DECREMENT TIMEOUT COUNT
006273  056215                  BNE     T10LOP                  ; IF UNEXPIRED, BRANCH
006274  103201  077674          BIC     #077674,R1              ; CLEAR UNUSED BITS
006276                          HARDER  29,R1                   ; REPORT ERROR
006276  104200  002007  001037                  MOV     #MS29,OUT.04
006301  104010  001040                          MOV     R1,OUT.05
006303  104300  001475  001036                  MOV     LUNIT,OUT.03
006306  104202  105725                          MOV     #29!ERHARD+3000.,R2
006310  104020  001035                          MOV     R2,OUT.02
006312  104200  006312  001034                  MOV     #.,OUT.01
006315  104200  060013  001033                  MOV     #ERRMES,OUT.RQ
006320  021740                  CALL    TESTEV


006321                  T12:
006321  104300  002760  002746          MOV     RUNCMD,RUN              ; RESTORE RUN COMMAND
006324  104200  000001  002726          MOV     #1,CR.RUN+1             ; RESTORE COMMAND LENGTH
006327  001617                          BR      TESTX                   ; BRANCH TO TEST NEXT UNIT
```

```
                                        .SBTTL  SNDLV1 AND TSTCMD
                                ; *** SEND LEVEL 1 START COMMAND
                                ;       SEND THE START FRAME AND RETURN
006330  101200  000014  002746  SNDLV1: BIS     #RUNLBC,RUN             ; SET RUN COMMAND
006333  104203  002725          SNDL1A: MOV     #CR.RUN,R3              ; SEND COMMAND
006335  104302  002646                  MOV     SDI,R2                  ; R2 -> PORT
006337  104237                          MOV     (R3)+,R0                ; SET UP PARAMETERS
006340  104131                          MOV     (R3),R1
006341  060004                          XFC     SEND                    ; SEND COMMAND
006342  000000                          RETURN

                                ; *** TEST THE COMMAND
                                ;       INPUT   RUN HAS LEVEL 1 OR 2 COMMAND SET UP
                                ;               CR.RUN = 0 TO INITIATE LEVEL 1 TRANMITTION
                                ;                      = WHATEVER FOR LEVEL 2
                                ;       OUTPUT  WILL NOT RETURN IF ERROR OCCURED
                                ;
006343  104203  002725          TSTCMD: MOV     #CR.RUN,R3              ; SEND COMMAND
006345  021076                  TSTCM2: CALL    TALK
006346  115002                          TST     R2                      ; DID IT FAIL?
006347  016357                          BEQ     T11ER                   ; IF NOT, BRANCH TO REPORT AND EXIT
006350  104203  002701                  MOV     #CR.GST,R3              ; CHECK STATUS
006352  021076                          CALL    TALK                    ; GET STATUS
006353  102200  000350  002766          BIT     #<ST.PE+ST.RE+ST.FE+ST.WE>,ST+ST.ERR
006356  056463                          BNE     TSTCME                  ; BRANCH IF ERROR
                                ; *** DID NOT FAIL -> ERROR
006357  115000  002647          T11ER:  TST     COMND                   ; DID WE HAVE COMMAND SET
006361  056437                          BNE     T11X2                   ; IF SO, LEVEL 1 XMIT ERROR
006362  106300  002760  002746          CMP     RUNCMD,RUN              ; ELSE, WHAT TYPE LEVEL 2 XMIT ERROR?
006365  056413                          BNE     T11X1                   ; IF RUN NOT SET TO RUN COMMAND, BRANCH
006366                                  HARDER  31,<R4,R5>
006366  104200  002077  001037                          MOV     #MS31,OUT.04
006371  104040  001040                  MOV     R4,OUT.05
006373  104050  001041                  MOV     R5,OUT.06
006375  104300  001475  001036          MOV     LUNIT,OUT.03
006400  104202  105727                  MOV     #31!ERHARD+3000.,R2
006402  104020  001035                  MOV     R2,OUT.02
006404  104200  006404  001034          MOV     #.,OUT.01
006407  104200  060013  001033          MOV     #ERRMES,OUT.RQ
006412  001753                  BR      TESTED
006413                  T11X1:  HARDER  30,RUN
006413  104200  002053  001037                  MOV     #MS30,OUT.04
006416  104300  002746  001040          MOV     RUN,OUT.05
006421  104300  001475  001036          MOV     LUNIT,OUT.03
006424  104202  105726                  MOV     #30!ERHARD+3000.,R2
006426  104020  001035                  MOV     R2,OUT.02
006430  104200  006430  001034          MOV     #.,OUT.01
006433  104200  060013  001033          MOV     #ERRMES,OUT.RQ
006436  001753                  BR      TESTED
006437                  T11X2:  HARDER  32,COMND
006437  104200  002130  001037                  MOV     #MS32,OUT.04
006442  104300  002647  001040          MOV     COMND,OUT.05
006445  104300  001475  001036          MOV     LUNIT,OUT.03
006450  104202  105730                  MOV     #32!ERHARD+3000.,R2
006452  104020  001035                  MOV     R2,OUT.02
006454  104200  006454  001034          MOV     #.,OUT.01
006457  104200  060013  001033          MOV     #ERRMES,OUT.RQ
```

```
     36 006462  001753                      BR      TESTED
     37 006463  021761            TSTCME:   CALL    DR.CLR          ; CLEAR ERROR
     38 006464  000000                      RETURN
```

```
                                        .SBTTL  FORTRK - FORMAT TRACK
006465                          FORTRK:
                                ;
                                ;       FORMAT TRACK GIVEN BY CURTRK, STARTING WITH DBN GIVEN IN TSTBLK
                                ;       THE DATA WRITTEN IS UNPREDICTABLE
                                ;
006465  104307  003074                  MOV     TSTBLK,R0               ; GET LO STARTING DBN
006467  104301  003011                  MOV     SUB+HIDBN,R1            ; GET HIGH ORDER BITS OF STARTING DBN
006471  110601                          ROR     R1                      ; MOVE TO CORRECT POSITION
006472  110601                          ROR     R1
006473  110601                          ROR     R1
006474  110601                          ROR     R1
006475  103201  170377                  BIC     #HBLONB,R1              ; CLEAR UNUSED BITS
006477  101201  140000                  BIS     #HD.DBN,R1              ; SET HEADER CODE
006501  101301  003075                  BIS     TSTBLK+1,R1             ; GET HI STARTING DBN
006503  104203  003256                  MOV     #OBUFF,R3               ; POINT TO OUTPUT BUFFER
006505  104304  003073                  MOV     SECTRK,R4               ; R4 CONTAINS NUMBER OF SECTORS TO FORMAT
006507  104205  003665                  MOV     #FCHAIN,R5              ; R5 POINTS TO FORMAT CHAIN
006511  100253                  FLOOP:  MOV     R3,(R5)+                ; MOVE POINTER TO BUFFER TO CHAIN
006512  100257                          MOV     R0,(R5)+                ; MOVE LO DBN TO CHAIN
006513  100251                          MOV     R1,(R5)+                ; MOVE HI DBN TO CHAIN
006514  115407                          INC     R0                      ; INCREMENT LO DBN
006515  046517                          BCC     FCARY                   ; IF NO CARRY, BRANCH
006516  115401                          INC     R1                      ; INCREMENT HI DBN
006517  117404                  FCARY:  DEC     R4                      ; DECREMENT SECTOR COUNT
006520  056511                          BNE     FLOOP                   ; BRANCH IF COUNT UNEXHAUSTED
006521  104207  100000                  MOV     #FSTOP,R0               ; GET FORMAT END-OF-CHAIN FLAG
006523  100157                          MOV     R0,(R5)                 ; MOVE INTO CHAIN
006524  104302  002646                  MOV     SDI,R2                  ; MOVE MASK TO R2
006526  020754                          CALL    RDSTAT                  ; GET DRIVE STATUS
006527  115002                          TST     R2                      ; WAS IT OK?
006530  016576                          BEQ     2$                      ; IF NO ERROR, BRANCH
006531  102201  000400                  BIT     #RCVERR,R1              ; SEE IF ANY ERRORS
006533  056555                          BNE     1$                      ; IF NOT, ERROR
006534                                  HARDER  10                      ; REPORT INVALID STATUS ERROR
006534  104200  000341  001037                  MOV     #MS10,OUT.04
006537  104300  001475  001036                  MOV     LUNIT,OUT.03
006542  104202  105702                          MOV     #10!ERHARD+3000.,R2
006544  104020  001035                          MOV     R2,OUT.02
006546  104200  006546  001034                  MOV     #.,OUT.01
006551  104200  060013  001033                  MOV     #ERRMES,OUT.RQ
006554  001732                          BR      TESTEW          ; BRANCH TO DONE
006555                          1$:
006555                                  HARDER  12                      ; REPORT XMIT ERROR
006555  104200  000434  001037                  MOV     #MS12,OUT.04
006560  104300  001475  001036                  MOV     LUNIT,OUT.03
006563  104202  105704                          MOV     #12!ERHARD+3000.,R2
006565  104020  001035                          MOV     R2,OUT.02
006567  104200  006567  001034                  MOV     #.,OUT.01
006572  104200  060013  001033                  MOV     #ERRMES,OUT.RQ
006575  001732                          BR      TESTEW                  ; BRANCH TO DONE
006576                          2$:
006576  102201  100000                  BIT     #RWRDY,R1               ; TEST R/W READY
006600  056622                          BNE     FGO                     ; IF ASSERTED, BRANCH
006601                                  HARDER  18                      ;READ/WRITE READY DROPPED BEFORE FORMAT
006601  104200  000737  001037                  MOV     #MS18,OUT.04
006604  104300  001475  001036                  MOV     LUNIT,OUT.03
```

```
                006607  104202  105712                                                  MOV     #18!ERHARD+3000.,R2
                006611  104020  001035                                                  MOV     R2,OUT.02
                006613  104200  006613  001034                                          MOV     #.,OUT.01
                006616  104200  060013  001033                                          MOV     #ERRMES,OUT.RQ
     44 006621  001740                          BR      TESTEV                  ; BRANCH TO EXIT
     45 006622  104207  003665          FGO:    MOV     #FCHAIN,R0              ; POINT TO FORMAT CHAIN (FOR XFC)
     46 006624  104304  003013                  MOV     SUB+DATPRE,R4           ; GET DATA PREAMBLE LENGTHS
     47 006626  103204  177400                  BIC     #HIBYTE,R4              ; CLEAR UNUSED BITS
     48 006630  104303  003013                  MOV     SUB+HDRPRE,R3           ; GET HEADER PREAMBLE LENGTH
     49 006632  110703                          SWAB    R3                      ; SWAP BYTES
     50 006633  103203  177400                  BIC     #HIBYTE,R3              ; CLEAR UNUSED BYTES
     51 006635  104301  003111                  MOV     CURTRK,R1               ; TRACK NUMBER
     60 006637  104302  002646                  MOV     SDI,R2                  ; SET PORT INDICATOR
     64 006641  060001                          XFC     FORMAT                  ; FORMAT THE TRACK (BUFFER CONTENTS ARE A DON'T CARE
     65 006642  115001                          TSTB    R1                      ; SEE IF ANY ERRORS OCCURRED
     66 006643  016707                          BEQ     FOREXT                  ; IF NOT, BRANCH
     67 006644                                  HARDER  19,<INS+1,INS+2,INS+3,CURTRK> ;TIMEOUT OF DRIVE OR READ/WRITE READY DROPPED
                006644  104200  000771  001037                                          MOV     #MS19,OUT.04
                006647  104300  002751  001040                                          MOV     INS+1,OUT.05
                006652  104300  002752  001041                                          MOV     INS+2,OUT.06
                006655  104300  002753  001042                                          MOV     INS+3,OUT.07
                006660  104300  003111  001043                                          MOV     CURTRK,OUT.08
                006663  104300  001475  001036                                          MOV     LUNIT,OUT.03
                006666  104202  105713                                                  MOV     #19!ERHARD+3000.,R2
                006670  104020  001035                                                  MOV     R2,OUT.02
                006672  104200  006672  001034                                          MOV     #.,OUT.01
                006675  104200  060013  001033                                          MOV     #ERRMES,OUT.RQ
     68 006700                                  ENDERR  19
                006700  104200  003276  001056                                          MOV     #SER22,OUT.19
                006703  104200  000024  003116                                          MOV     #19+1,ERRPOS    ; SET THE POSITION
     69 006706  021746                          CALL    TESTEX
     70                                 ;       MOV     OUT.RQ,R0               ; PRINT ERROR
     71                                 ;       CALL    HOSTRQ
     72 006707  000000                  FOREXT: RETURN
```

```
                                                .SBTTL  TRKTST - TEST TRACK
006710                                  TRKTST:
                                        ;
                                        ;       TEST THE ENTIRE TRACK UNTIL AT LEAST ONE BLOCK IS SUCCESSFULLY
                                        ;       WRITTEN AND READ WITH SEVERAL DATA PATTERNS
                                        ;
006710  104307  003074                        MOV     TSTBLK,R0               ; MOVE LO STARTING DBN TO CURRENT DBN
006712  104070  003104                        MOV     R0,CURBLK
006714  104307  003075                        MOV     TSTBLK+1,R0             ; MOVE HI STARTING DBN TO CURRENT DBN
006716  104070  003105                        MOV     R0,CURBLK+1
006720  104301  003073                        MOV     SECTRK,R1               ; GET SECTORS/TRACK
006722  117401                              DEC     R1                      ; ADJUST FOR END LOOP WHEN NEGATIVE
006723  104010  003113              TRKLOP:   MOV     R1,SECCNT               ; SAVE SECTORS LEFT IN SECTOR COUNT
006725  104201  000001                        MOV     #1,R1                   ; MOVE 1 TO CURRENT PATTERN
006727  026752                              CALL    WTRCMP                  ; WRITE THAN READ AND COMPARE THE SECTOR
006730  115007                              TST     R0                      ; SEE IF WRITES, READS AND COMPARES WERE GOOD
006731  016751                              BEQ     TRKEXT                  ; IF SO, BRANCH
006732  104307  003104                        MOV     CURBLK,R0               ; GET LO CURRENT BLOCK NUMBER
006734  115407                              INC     R0                      ; INCREMENT DBN NUMBER
006735  104070  003104                        MOV     R0,CURBLK               ; SAVE
006737  046745                              BCC     TRKBOT                  ; BRANCH IF NO CARRY
006740  104307  003105                        MOV     CURBLK+1,R0             ; GET HI DBN NUMBER
006742  115407                              INC     R0                      ; INCREMENT
006743  104070  003105                        MOV     R0,CURBLK+1             ; SAVE
006745  104301  003113              TRKBOT:   MOV     SECCNT,R1               ; GET SECTOR COUNT
006747  117401                              DEC     R1                      ; DECREMENT COUNT
006750  036723                              BPL     TRKLOP                  ; BRANCH IF COUNT POSITIVE
006751  000000                  TRKEXT: RETURN
```

```
                                                .SBTTL  WRTCMP - WRITE A DATA PATTERN AND COMPARE
006752                                  WTRCMP:
                                        ;
                                        ;       WRITE A DATA PATTERN TO A SECTOR, READ IT BACK, THEN DO A DATA
                                        ;       COMPARE ON THE DATA.  DO THIS AS MANY TIMES AS THERE IS PATTERNS.
                                        ;       CURRENT PATTERN NUMBER IN 'CURPAT'
                                        ;
006752  104010  003112                          MOV     R1,CURPAT               ; R1 IS CURRENT PATTERN NUMBER
006754  105201  003145                          ADD     #PATPTR,R1              ; R1 NOW POINTS TO PATTERN TO GENERATE
006756  026777                                  CALL    GENPAT                  ; GENERATE THE PATTERN IN THE OUTPUT BUFFER
006757  027020                                  CALL    WRITEB                  ; WRITE THE PATTERN
006760  115007                                  TST     R0                      ; SEE IF ANY ERROR OCCURRED
006761  056776                                  BNE     WTREXT                  ; IF SO, BRANCH
006762  027265                                  CALL    READB                   ; READ THE BLOCK BACK
006763  115007                                  TST     R0                      ; SEE IF ANY ERRORS OCCURRED
006764  056776                                  BNE     WTREXT                  ; IF SO, BRANCH
006765  027504                                  CALL    CMPDAT                  ; COMPARE THE DATA
006766  115007                                  TST     R0                      ; SEE IF ANY ERRORS OCCURRED
006767  056776                                  BNE     WTREXT                  ; IF SO, BRANCH
006770  104301  003112                          MOV     CURPAT,R1               ; GET CURRENT PATTERN
006772  115401                                  INC     R1                      ; NEXT PATTERN
006773  106301  003145                          CMP     PATPTR,R1               ; COMARE AGAINST MAXIMUM PATTERN NUMBER
006775  036752                                  BPL     WTRCMP                  ; IF LESS OR EQUAL, LOOP
006776  000000                  WTREXT: RETURN
```

```
      1
      2 006777                          GENPAT:   .SBTTL  GENERATE A PATTERN
      3                                           ;
      4                                           ;       GENERATE THE DATA PATTERN IN THE OUTPUT BUFFER
      5                                           ;
      6                                           ;       R1 POINTS TO POINTER TO PATTERN BUFFER WHICH IS PATTERN LENGTH
      7                                           ;       (1 WORD) FOLLOWED BY THAT MANY WORDS OF PATTERN
      8                                           ;
      9 006777                                    PUSH    R2                      ; SAVE R2
        006777  100462                                                                    MOV R2,-(SP)
     10 007000  104117                            MOV     (R1),R0                 ; R0 POINTS TO START OF PATTERN BUFFER
     11 007001  104203  000400                     MOV     #256.,R3                ; R3 HOLDS PATTERN COUNT
     12 007003  104204  003256                     MOV     #OBUFF,R4               ; R4 POINTS TO OUTPUT BUFFER
     13 007005  104071                  GENOUT:    MOV     R0,R1                   ; R1 POINTS TO START OF PATTERN BUFFER
     14 007006  104215                            MOV     (R1)+,R5                ; R5 CONTAINS LENGTH OF PATTERN
     15 007007  104212                  GENIN:    MOV     (R1)+,R2                ; GET WORD OF PATTERN
     16 007010  100242                            MOV     R2,(R4)+                ; MOVE TO BUFFER
     17 007011  117403                            DEC     R3                      ; DECREMENT BUFFER COUNT
     18 007012  017016                            BEQ     GENEXT                  ; IF BUFFER FULL, EXIT
     19 007013  117405                            DEC     R5                      ; DECREMENT PATTERN COUNT
     20 007014  057007                            BNE     GENIN                   ; IF NON-ZERO, BRANCH
     21 007015  007005                            BR      GENOUT                  ; START PATTERN OVER AGAIN
     22 007016                          GENEXT:    POP     R2                      ; RESTORE R2
        007016  104262                                                                    MOV (SP)+,R2
     23 007017  000000                            RETURN
```

```
                                        .SBTTL  WRITEB - WRITE A SECTOR
007020                          WRITEB:
                                ;
                                ;       WILL WRITE ONE SECTOR OF DATA TO THE DISK
                                ;
007020  104207  140000                  MOV     #WSTOP,R0
007022  104070  003137                  MOV     R0,RW.STAT+CHAIN; MOVE END-OF-CHAIN TO CHAIN
007024  104307  003104                  MOV     CURBLK,R0               ; MOVE LO DBN TO CHAIN
007026  104070  003141                  MOV     R0,RW.LOW+CHAIN
007030  104307  003105                  MOV     CURBLK+1,R0             ; MOVE HI DBN TO CHAIN
007032  104301  003011                  MOV     SUB+HIDBN,R1            ; GET HIGH ORDER BITS OF STARTING DBN
007034  110601                          ROR     R1                      ; MOVE TO CORRECT POSITION
007035  110601                          ROR     R1
007036  110601                          ROR     R1
007037  110601                          ROR     R1
007040  103201  170377                  BIC     #HBLONB,R1              ; CLEAR UNUSED BITS
007042  101201  140000                  BIS     #HD.DBN,R1              ; SET HEADER CODE
007044  101017                          BIS     R1,R0                   ; SET IN R0
007045  104070  003142                  MOV     R0,RW.HI+CHAIN
007047  104307  003111                  MOV     CURTRK,R0               ; MOVE TRACK + RTC TO CHAIN
007051  101207  122400                  BIS     #WREAL,R0               ; SET REAL TIME COMMAND WRITTE
007053  104070  003143                  MOV     R0,RW.CMD+CHAIN
007055  104207  003256                  MOV     #OBUFF,R0               ; MOVE OUTPUT BUFFER TO CHAIN
007057  104070  003140                  MOV     R0,RW.BUF+CHAIN
007061  104302  002646                  MOV     SDI,R2                  ; MOVE MASK TO R2
007063  020754                          CALL    RDSTAT                  ; GET DRIVE STATUS
007064  115002                          TST     R2                      ; WAS IT OK?
007065  017133                          BEQ     2$                      ; IF NO ERROR, BRANCH
007066  102201  000400                  BIT     #RCVERR,R1              ; SEE IF ANY ERRORS
007070  057112                          BNE     1$                      ; IF NOT, ERROR
007071                                  HARDER  10                      ; REPORT INVALID STATUS ERROR
007071  104200  000341  001037                  MOV     #MS10,OUT.04
007074  104300  001475  001036                  MOV     LUNIT,OUT.03
007077  104202  105702                          MOV     #10!ERHARD+3000.,R2
007101  104020  001035                          MOV     R2,OUT.02
007103  104200  007103  001034                  MOV     #.,OUT.01
007106  104200  060013  001033                  MOV     #ERRMES,OUT.RQ
007111  001732                          BR      TESTEW                  ; BRANCH TO DONE
007112                          1$:
007112                                  HARDER  12                      ; REPORT XMIT ERROR
007112  104200  000434  001037                  MOV     #MS12,OUT.04
007115  104300  001475  001036                  MOV     LUNIT,OUT.03
007120  104202  105704                          MOV     #12!ERHARD+3000.,R2
007122  104020  001035                          MOV     R2,OUT.02
007124  104200  007124  001034                  MOV     #.,OUT.01
007127  104200  060013  001033                  MOV     #ERRMES,OUT.RQ
007132  001732                          BR      TESTEW                  ; BRANCH TO DONE
007133                          2$:
007133  102201  100000                  BIT     #RWRDY,R1               ; SEE IF READ/WRITE READY IS ASSERTED
007135  057164                          BNE     WGO                     ; IF SO, BRANCH
007136  104307  003113                  MOV     SECCNT,R0               ; GET SECTOR COUNT
007140  057264                          BNE     WRTEXT                  ; IF NONZERO, DO NOT REPORT ERROR
007141                                  HARDER  22                      ;READ/WRITE DROPPED READY BEFORE WRITE
007141  104200  001117  001037                  MOV     #MS22,OUT.04
007144  104300  001475  001036                  MOV     LUNIT,OUT.03
007147  104202  105716                          MOV     #22!ERHARD+3000.,R2
007151  104020  001035                          MOV     R2,OUT.02
```

```
                                                                    MOV     #.,OUT.01
                                                                    MOV     #ERRMES,OUT.RQ
 49 007161  104207  000001                MOV     #1,R0                   ; FLAG ERROR
 50 007163  007264                        BR      WRTEXT                  ; BRANCH
 51 007164  104302  002646          WGO:  MOV     SDI,R2                  ; SET PORT INDICATOR
 52 007166  060012                        XFC     WAITSI                  ; WAIT FOR SECTOR OR INDEX PULSE
 53 007167  104207  003137                MOV     #CHAIN,R0               ; POINT TO WRITE CHAIN
 54 007171  104304  003013                MOV     SUB+DATPRE,R4           ; MOVE DATA PREAMBLE LENGTH TO R4
 55 007173  103204  177400                BIC     #HIBYTE,R4              ; CLEAR UNUSED BITS
 63 007175  060003                        XFC     XWRITE                  ; WRITE THE BLOCK
 64 007176  114007                        CLR     R0                      ; FLAG AS NO ERRORS
 65 007177  115000  003117                TST     WFLAG                   ; DID WE TEST WHEN WRITE PROTECTED?
 66 007201  057264                        BNE     WRTEXT                  ; IF SO, EXIT
 67 007202  115001                        TST     R1                      ; SEE IF AN ERROR OCCURRED
 68 007203  017264                        BEQ     WRTEXT                  ; IF NOT, BRANCH
 69 007204  104307  003113                MOV     SECCNT,R0               ; SEE IF ERROR SHOULD BE REPORTED
 70 007206  057264                        BNE     WRTEXT                  ; IF NOT, BRANCH
 71 007207                                HARDER  23,<R1,CURBLK,CURBLK+1,INS+1,INS+2,INS+3,CURTRK> ;ERROR DURING WRITE
    007207  104200  001151  001037                                        MOV     #MS23,OUT.04
    007212  104010  001040                                                MOV     R1,OUT.05
    007214  104300  003104  001041                                        MOV     CURBLK,OUT.06
    007217  104300  003105  001042                                        MOV     CURBLK+1,OUT.07
    007222  104300  002751  001043                                        MOV     INS+1,OUT.08
    007225  104300  002752  001044                                        MOV     INS+2,OUT.09
    007230  104300  002753  001045                                        MOV     INS+3,OUT.10
    007233  104300  003111  001046                                        MOV     CURTRK,OUT.11
    007236  104300  001475  001036                                        MOV     LUNIT,OUT.03
    007241  104202  105717                                                MOV     #23!ERHARD+3000.,R2
    007243  104020  001035                                                MOV     R2,OUT.02
    007245  104200  007245  001034                                        MOV     #.,OUT.01
    007250  104200  060013  001033                                        MOV     #ERRMES,OUT.RQ
 72 007253                                ENDERR  12
    007253  104200  003276  001047                                        MOV     #SER22,OUT.12
    007256  104200  000015  003116                                        MOV     #12+1,ERRPOS    ; SET THE POSITION
 73 007261  021746                        CALL    TESTEX
 74                                 ;     MOV     OUT.RQ,R0
 75                                 ;     CALL    HOSTRQ
 76 007262  104207  000001                MOV     #1,R0                   ; MAKE R0 NONZERO TO REPORT ERROR
 77 007264  000000                  WRTEXT: RETURN
```

```
                                                .SBTTL  READB - READ ONE SECTOR
007265                                  READB:
                                        ;
                                        ;       READ ONE SECTOR FROM DISK
                                        ;
007265  104207  100000                  ;       MOV     #RSTOP,R0               ; MOVE END-OF-CHAIN TO CHAIN
007267  104070  003137                  MOV     R0,RW.STAT+CHAIN
007271  104307  003111                  MOV     CURTRK,R0               ; MOVE TRACK NUMBER AND RTC TO CHAIN
007273  101207  013400                  BIS     #RREAL,R0               ; SET REAL TIME READ COMMMAND
007275  104070  003143                  MOV     R0,RW.CMD+CHAIN
007277  104207  003665                  MOV     #RBUFD,R0               ; MOVE POINTER TO INPUT BUFFER INTO CHAIN
007301  104070  003140                  MOV     R0,RW.BUF+CHAIN
007303  104302  002646                  MOV     SDI,R2                  ; MOVE MASK TO R2
007305  020754                          CALL    RDSTAT                  ; GET DRIVE STATUS
007306  115002                          TST     R2                      ; WAS IT OK?
007307  017355                          BEQ     2$                      ; IF NO ERROR, BRANCH
007310  102201  000400                  BIT     #RCVERR,R1              ; SEE IF ANY ERRORS
007312  057334                          BNE     1$                      ; IF NOT, ERROR
007313                                  HARDER  10                      ; REPORT INVALID STATUS ERROR
007313  104200  000341  001037                          MOV     #MS10,OUT.04
007316  104300  001475  001036                          MOV     LUNIT,OUT.03
007321  104202  105702                          MOV     #10!ERHARD+3000.,R2
007323  104020  001035                          MOV     R2,OUT.02
007325  104200  007325  001034                          MOV     #.,OUT.01
007330  104200  060013  001033                          MOV     #ERRMES,OUT.RQ
007333  001732                          BR      TESTEW                  ; BRANCH TO DONE
007334                                  1$:
007334                                  HARDER  12                      ; REPORT XMIT ERROR
007334  104200  000434  001037                          MOV     #MS12,OUT.04
007337  104300  001475  001036                          MOV     LUNIT,OUT.03
007342  104202  105704                          MOV     #12!ERHARD+3000.,R2
007344  104020  001035                          MOV     R2,OUT.02
007346  104200  007346  001034                          MOV     #.,OUT.01
007351  104200  060013  001033                          MOV     #ERRMES,OUT.RQ
007354  001732                          BR      TESTEW                  ; BRANCH TO DONE
007355                                  2$:
007355  102201  100000                  BIT     #RWRDY,R1               ; SEE IF READ/WRITE READY IS ASSERTED
007357  057412                          BNE     RGO                     ; IF SO, BRANCH
007360  104307  003113                  MOV     SECCNT,R0               ; GET SECTOR COUNT
007362  057503                          BNE     REDEXT                  ; IF NONZERO, DO NOT REPORT ERROR
007363                                  HARDER  24                      ;READ/WRITE DROPPED READY BEFORE READ
007363  104200  001306  001037                          MOV     #MS24,OUT.04
007366  104300  001475  001036                          MOV     LUNIT,OUT.03
007371  104202  105720                          MOV     #24!ERHARD+3000.,R2
007373  104020  001035                          MOV     R2,OUT.02
007375  104200  007375  001034                          MOV     #.,OUT.01
007400  104200  060013  001033                          MOV     #ERRMES,OUT.RQ
007403                                  ENDERR  5
007403  104200  003276  001040                          MOV     #SER22,OUT.05
007406  104200  000006  003116                          MOV     #5+1,ERRPOS     ; SET THE POSITION
007411  007500                          BR      REDERR                  ; GO PRINT THE ERROR
007412  104302  002646          RGO:    MOV     SDI,R2                  ; SET PORT INDICATOR
007414  060012                          XFC     WAITSI                  ; WAIT FOR SECTOR OR INDEX PULSE
007415  104207  003137                  MOV     #CHAIN,R0               ; POINT TO READ CHAIN
007417  060002                          XFC     XREAD                   ; READ THE SECTOR
007420  114007                          CLR     R0                      ; FLAG AS NO ERRORS
007421  115001                          TST     R1                      ; SEE IF ERROR OCCURRED
```

```
     52 007422  017503                                BEQ     REDEXT                  ; IF NOT, BRANCH
     53 007423  104307  003113                        MOV     SECCNT,R0               ; SEE IF ERROR SHOULD BE REPORTED
     54 007425  057503                                BNE     REDEXT                  ; IF NOT, BRANCH
     55 007426                                        HARDER  25,<R1,CURBLK,CURBLK+1,INS+1,INS+2,INS+3,CURTRK> ;ERROR DURING READ
        007426  104200  001337  001037                                                  MOV     #MS25,OUT.04
        007431  104010  001040                                                          MOV     R1,OUT.05
        007433  104300  003104  001041                                                  MOV     CURBLK,OUT.06
        007436  104300  003105  001042                                                  MOV     CURBLK+1,OUT.07
        007441  104300  002751  001043                                                  MOV     INS+1,OUT.08
        007444  104300  002752  001044                                                  MOV     INS+2,OUT.09
        007447  104300  002753  001045                                                  MOV     INS+3,OUT.10
        007452  104300  003111  001046                                                  MOV     CURTRK,OUT.11
        007455  104300  001475  001036                                                  MOV     LUNIT,OUT.03
        007460  104202  105721                                                          MOV     #25!ERHARD+3000.,R2
        007462  104020  001035                                                          MOV     R2,OUT.02
        007464  104200  007464  001034                                                  MOV     #.,OUT.01
        007467  104200  060013  001033                                                  MOV     #ERRMES,OUT.RQ
     56 007472                                        ENDERR  12
        007472  104200  003276  001047                                                  MOV     #SER22,OUT.12
        007475  104200  000015  003116                                                  MOV     #12+1,ERRPOS    ; SET THE POSITION
     57 007500  021746                        REDERR: CALL    TESTEX
     58                                       ;       MOV     OUT.RQ,R0
     59                                       ;       CALL    HOSTRQ
     60 007501  104207  000001                        MOV     #1,R0                   ; FLAG ERROR
     61 007503  000000                        REDEXT: RETURN
```

```
                                                .SBTTL  CMPDAT - COMPARE DATA
007504                                  CMPDAT:
                                        ;
                                        ;       COMPARE THE DATA IN 'OBUFF' (OUTPUT BUFFER) WITH 'RBUFD'
                                        ;       (INPUT BUFFER)
                                        ;
007504  104207  003256                          MOV     #OBUFF,R0               ; R0 POINTS AT OUTPUT BUFFER
007506  104201  003665                          MOV     #RBUFD,R1               ; R1 POINTS AT INPUT BUFFER
007510  114003                                  CLR     R3                      ; COUNTER
007511  104274                          CMPLOP: MOV     (R0)+,R4                ; GET OUTPUT BUFFER WORD
007512  106214                                  CMP     (R1)+,R4                ; COMPARE AGAINST INPUT BUFFER
007513  017567                                  BEQ     NOERR                   ; IF NO ERROR, BRANCH
007514  104305  003113                          MOV     SECCNT,R5               ; SEE IF ERROR IS TO BE REPORTED
007516  057574                                  BNE     CMPEXT                  ; IF NOT, BRANCH
007517                                          HARDER  26,<R3,-(R0),-(R1),INS+1,INS+2,INS+3,CURTRK> ;DATA COMPARE FAILURE
007517  104200  001473  001037                                                          MOV     #MS26,OUT.04
007522  104030  001040                                                                  MOV     R3,OUT.0
007524  104470  001041                                                                  MOV     -(R0),OU  )6
007526  104410  001042                                                                  MOV     -(R1),OU. 07
007530  104300  002751  001043                                                          MOV     INS+1,OUT.08
007533  104300  002752  001044                                                          MOV     INS+2,OUT.09
007536  104300  002753  001045                                                          MOV     INS+3,OUT.10
007541  104300  003111  001046                                                          MOV     CURTRK,OUT.11
007544  104300  001475  001036                                                          MOV     LUNIT,OUT.03
007547  104202  105722                                                                  MOV     #26!ERHARD+3000.,R2
007551  104020  001035                                                                  MOV     R2,OUT.02
007553  104200  007553  001034                                                          MOV     #.,OUT.01
007556  104200  060013  001033                                                          MOV     #ERRMES,OUT.RQ
007561  104307  001033                          MOV     OUT.RQ,R0
007563  021000                                  CALL    HOSTRQ
007564  104207  000001                          MOV     #1,R0                   ; SET FOR ERROR
007566  007574                                  BR      CMPEXT                  ; EXIT
007567  115403                          NOERR:  INC     R3                      ; INCREMENT COUNT
007570  106203  000400                          CMP     #256.,R3                ; SEE IF COMPARE COMPLETE
007572  057511                                  BNE     CMPLOP                  ; IF NOT, BRANCH
007573  114007                                  CLR     R0                      ; FLAG AS NO ERRORS
007574  000000                          CMPEXT: RETURN
                                        ;
001320                                  OVL.TS  =       .-T7
007575                                          DMEND
007575  000105                                  .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
000001                                          .END
```

SYMBOL TABLE

```
AREA    003114          DRVRUN= 000014          GRPOFF= 000011          MS.CNT= 152000          OCNT$ = 000003
ATTN  = 000002          DR.CLR  001761          GST     002745          MS.DIS  002621          OLDGRP  003122
AVAIL = 000100          ECC   = 000015          HBHINB= 007777          MS.END= 131000          ONL     002732
BF.DAT= 000000          ECCFLG= 010000          HBLONB= 170377          MS.GCR  002630          OTABLE  001547
BF.ECC= 000401          ECCRSH= 000002          HDRPRE= 000005          MS.GST  002667          OUT.RQ  001033
BF.EDC= 000400          ECHO  = 000010          HD.BAD= 110000          MS.INR  002712          OUT.01  001034
BLOCKC  003103          ECHOC = 000350          HD.DBN= 140000          MS.MOD  002676          OUT.02  001035
BLOCKG  003102          ENDGTU  004306          HD.LBN= 000000          MS.ONL  002604          OUT.03  001036
BLOCKT  003101          ENDPNT= 005262          HD.PRV= 050000          MS.RUN  002726          OUT.04  001037
BREAK = 000000          EOC   = 100000          HD.RBN= 060000          MS.SCR  002647          OUT.05  001040
BUFFLG= 040000          ERECOV= 000006          HD.REV= 030000          MS.SEK  002706          OUT.06  001041
BUFFSZ= 000400          ERHARD= 100000          HD.XBN= 120000          MS.STR= 070400          OUT.07  001042
BUFSIZ= 000043          ERLEV = 000002          HEADER= 000002          MS1     000000          OUT.08  001043
CHAIN   003137          ERRMC = 060014          HIBYTE= 177400          MS10    000341          OUT.09  001044
CHECK = 000010          ERRMES= 060013          HICYL = 000001          MS11    000366          OUT.10  001045
CHGMOD= 000201          ERRN  = 005670          HIDBN = 000003          MS12    000434          OUT.11  001046
CHRRES= 000170          ERRORS  002741          HILBN = 000002          MS13    000461          OUT.12  001047
CMPDAT  007504          ERRPOS  003116          HIMEM = 007774          MS14    000501          OUT.13  001050
CMPEXT  007574          ERSOFT= 140000          HIRBN = 000003          MS15    000555          OUT.14  001051
CMPLOP  007511          EXIT  = 000021          HIXBN = 000002          MS16    000635          OUT.15  001052
COMND   002647          FB.DAT= 000000          HOSTRQ  001000          MS17    000677          OUT.16  001053
COMPAR= 000006          FB.EDC= 000400          HRDREV= 000003          MS18    000737          OUT.17  001054
COMPLT= 000176          FCARY   006517          INDEXS  003136          MS19    000771          OUT.18  001055
COPY    002761          FCHAIN  003665          INR     002747          MS2     000014          OUT.19  001056
CR      002773          FCTSIZ= 000010          INS     002750          MS20    001050          OUT.20  001057
CR.CLR  002662          FDIACY  003067          INSEEK= 000012          MS2000  002734          OUT.21  001060
CR.DIS  002667          FGO     006622          INTEDC= 000105          MS21    001073          OUT.22  001061
CR.GCR  002655          FLAG    003120          IRECLB= 000216          MS22    001117          OUT.23  001062
CR.GST  002701          FLOOP   006511          LARGE = 000001          MS23    001151          OUT.24  001063
CR.INR  002720          FNDCYL  001770          LBHINB= 177417          MS24    001306          OUT.25  001064
CR.MOD  002706          FNDCY2  001776          LBLONB= 177760          MS25    001337          OUT.26  001065
CR.ONL  002650          FND3    002006          LBNCYL= 000000          MS26    001473          OUT.27  001066
CR.RUN  002725          FND4    002034          LBNHST= 000012          MS27    001633          OUT.28  001067
CR.SCR  002674          FOREXT  006707          LBNTRK= 000011          MS28    001716          OUT.29  001070
CR.SEK  002713          FORMAT= 000001          LCDEN   003062          MS29    002007          OUT.30  001071
CURBLK  003104          FORTRK  006465          LDIACY  003064          MS3     000032          OUT.31  001072
CURGRP  003110          FOR.SZ= 001375          LETTER  003115          MS30    002053          OUT.32  001073
CURPAT  003112          FRAME = 000004          LINIT   001476          MS31    002077          OUT.33  001074
CURTRK  003111          FSTOP = 100000          LINKLN= 000007          MS32    002130          OUT.34  001075
CVT   = 000020          FTLDEV= 040000          LOBYTE= 000377          MS33    002156          OVERFL= 000002
CYLLO   003131          FTLSYS= 000000          LONG  = 002717          MS34    002220          OVE.MN= 000714
C2HARD= 040000          FT.BUF= 000000          LONGTO= 000001          MS35    002245          OVE.MS= 000000
DATPRE= 000005          FT.HI = 000001          LOW   = 000002          MS36    002306          OVE.SU= 005262
DBNCYL= 000022          FT.LOW= 000002          LUNIT   001475          MS37    002331          OVE.TS= 005262
DCLOCK= 000004          FXBNCY  003071          MAXSND= 001750          MS38    002422          OVL.MN= 004347
DIA     002734          GCR     002742          MEDTYP= 000006          MS4     000063          OVL.MS= 003673
DINIT = 000011          GENEXT  007016          MESSAG= 060015          MS5     000110          OVL.SU= 001216
DIS     002736          GENIN   007007          MICREV= 000003          MS6     000135          OVL.TS= 002314
DISCON= 000204          GENOUT  007005          MOD     002754          MS7     000160          OVL...= 013774
DMSDI   003032          GENPAT  006777          MODE    002755          MS8     000221          OVRLAY  001633
DONE  = 060016          GETCHR= 000207          MODE1   002756          MS9     000306          OVRLNM  001730
DONECD  001627          GETSTA= 000011          MODE2   002757          MWR   = 000017          OVRLPT= 005262
DRC     002740          GETSUB= 000210          MOD512= 126736          NOERR   007567          OVRPNT  001731
DRTYPE= 000007          GETU    003665          MOD576= 074161          NSCSL   003033          OVSTRT= 007774
DRVCLR= 000005          GROUP   003133          MRD   = 000016          NUMOVL= 000003          OVS.MN= 001040
DRVID = 000004          GRPCNT  003121          MSSG$ = 000000          NUMPTR= 000014          OVS.MS= 011756
DRVONL= 000213          GRPCYL= 000002          MS.CLR  002611          OBUFF   003256          OVS.SU= 021544
```

SYMBOL TABLE

```
OVS.TS= 024200          RW.HI = 000003          SETUP = 000001          TEST4 = 000000          T6E     006241
OV...  = 014414          RW.LOW= 000002          SHRTTO= 000000          TIMEOU= 000001          T6F     006255
PATPTR  003145          RW.SDI= 000005          SL.GRP= 107000          TO      001463          T6G     006215
PAT4    003152          RW.STA= 000000          SNDAGN  001010          TOOBIG= 000001          T7      006255
PAT5    003173          SBCRES= 000167          SNDLV1  006330          TRACK   003134          T7A     006270
PAT6    003214          SCR     002743          SNDL1A  006333          TRKBOT  006745          T7C     006327
PAT8    003235          SCTWRD= 000377          SNDONE= 002666          TRKEXT  006751          T7X     006476
PHYSA = 001000          SDI     002646          SS     = 000001          TRKGRP= 000003          T8      005337
PORT2   001556          SDILTO  001462          SSCTOR  003135          TRKLOP  006723          T8A     005402
PORT3   001606          SDISTO  001461          ST      002764          TRKTST  006710          T8B     005420
PORT4   001614          SDIVER= 000000          STACK   001542          TSTBLK  003074          T8D     005452
PORT5   001623          SECCNT  003113          START   003665          TSTCMD  006343          T8E     005501
RBNTRK= 000004          SECCYL  003107          STATUS= 000007          TSTCME  006463          T8F     005562
RBUFD   003665          SECGRP  003106          STRST   002131          TSTCM2  006345          T8T     005472
RBUFLN= 000415          SECTRK  003073          STRTST  005262          TSTCYL  003076          T9      006203
RBUF0   003657          SEEK    002156          STSRES= 000366          T0A     005415          UDADM3= 001000 G
RCONT = 000000          SEEKA   002171          ST.C   = 000002          T0B     005422          UDA50 = 000001
RCTCPS= 000001          SEEK1   002203          ST.CON= 000002          T0B     005362          UDA52 = 000000
RCTCSZ= 000014          SEEK2   002204          ST.DB = 001000          T1MSIZ= 060000          UNITNB  002645
RCV    = 000005          SEEK3   002325          ST.DF = 000020          T10     006214          UNITS   002625
RCVERR= 000400          SEND   = 000004          ST.DR = 000040          T10LOP  006215          UNIT0 = 000001
RCVRDY= 000001          SERRTY  002763          ST.EL = 000010          T11     005563          UNIT1 = 000002
RDSTAT  000754          SER00   002573          ST.ERR= 000002          T11ER   006357          UNIT2 = 000004
READB   007265          SER10   003052          ST.FE = 000200          T11X1   006413          UNIT3 = 000010
READ1   002331          SER11   003065          ST.FO = 002000          T11X2   006437          UNSSUC= 000175
READ1A  002416          SER12   003103          ST.MOD= 000001          T12     006321          UREAD = 000013
READ1B  002542          SER13   003115          ST.MSK= 000000          T2CMD = 060002          UTOTST= 060012
READ1C  002521          SER14   003130          ST.OA = 000200          T2DLL = 060001          UWRITE= 000014
READ1E  002543          SER15   003156          ST.PE = 000040          T3A     005554          U.SUBU= 000050
READ1X  002617          SER16   003200          ST.PS = 000002          T3B     005603          U.UNUM= 000063
REDERR  007500          SER17   003223          ST.RE = 000100          T3C     005500          VAR1    003123
REDEXT  007503          SER18   003255          ST.RR = 000100          T3D     005604          VAR2    003125
RETRY   002762          SER18A  003274          ST.RTY= 000003          T3STRT  001555          VAR3    003127
RETS   = 000001          SER18B  003271          ST.RU = 000001          T4A     005632          VAR4    003130
REVECT= 000004          SER18C  003266          ST.SR = 000020          T4A1    005675          WAITSI= 000012
REVS   = 000007          SER18D  003263          ST.STA= 000001          T4B     005731          WBUFLN= 000401
RGO     007412          SER18E  001543          ST.S7 = 000400          T4BB1 = 060005          WCONT = 040000
RM     = 000004          SER22   003276          ST.UNT= 000000          T4BB2 = 060006          WFLAG   003117
ROFDBN  003055          SER23   003344          ST.WE = 000010          T4C     005764          WGO     007164
ROFDC   003057          SER36   002446          SUB    = 003006          T4D     005770          WREAL = 122400
RREAL = 013400          SER39   002532          SUBUNT  002744          T4MPRM= 060003          WRITEB  007020
RSTOP = 100000          SER40   003360          T       005332          T4MXFR= 060011          WRONG = 000002
RTDS    000720          SER41   003402          TALK    001076          T4SEEK= 060010          WRTEXT  007264
RTDSL   000746          SER50   003415          TEST   = 000002          T4SOFT= 060007          WSTOP = 140000
RUN     002746          SER51   003422          TESTED  001753          T4UPRM= 060004          WTRCMP  006752
RUNCMD  002760          SER52   003430          TESTEV  001740          T5      005715          WTREXT  006776
RUNLBC= 000014          SER53   003467          TESTEW  001732          T6      006003          XBNCYL= 000021
RWRDY = 100000          SER54   003520          TESTEX  001746          T6A     006012          XFERRT= 000000
RW.ANG= 000006          SER55   003551          TESTEY  001752          T6C     006052          XREAD = 000002
RW.BUF= 000001          SER56   003604          TESTX   001617          T6D     006056          XWRITE= 000003
RW.CMD= 000004          SER57   003634

. ABS.  031030      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  8789 WORDS  ( 35 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  71 PAGES
```

,B:UDAT3.L50/C=$DMACRO,B:UDAT3

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
LOW      6-70#
LUNIT    22-8     23-13    25-23    25-43    25-47    25-51    25-55    25-59    25-64    25-71    26-16#   28-22*   29-26    33-29
         33-33    33-41    34-60    34-64    34-70    34-96    40-38    43-37    47-29    47-38    47-44    49-37    50-46    50-53
         51-75    54-19    55-94    58-111   58-117   58-119   59-17    59-20    59-28    60-31    60-33    60-35    61-35    61-38
         61-43    61-67    65-38    65-41    65-48    65-71    66-26    66-29    66-36    66-55    67-15
MAXSND    4-71#   25-13    37-80
MEDTYP    6-35#
MESSAG    7-16#   47-38
MICREV    6-12#
MOD      35-25    35-51#
MOD512    5-24#   58-113
MOD576    5-25#   58-115
MODE     35-52#   46-5*    55-8*
MODE1    35-53#   46-5
MODE2    35-54#   55-8
MRD       5-17#   23-17
MS.CLR   41-1#    45-26
MS.CNT    4-87#   57-26
MS.DIS   41-1#    59-3
MS.END    4-86#   57-10    57-14    57-22    57-29
MS.GCR   41-1#    44-6
MS.GST   41-1#    45-5     47-22
MS.INR   41-1#    47-5
MS.MOD   41-1#    46-6     55-9
MS.ONL   41-1#    44-32
MS.RUN   41-1#    46-19
MS.SCR   41-1#    48-13
MS.SEK   33-16    41-1#
MS.STR    4-85#   57-5     57-19
MS1      25-23    41-1#
MS10     33-29    34-60    41-1#    61-35    65-38    66-26
MS11     41-1#    59-17
MS12     33-33    34-64    41-1#    59-20    61-38    65-41    66-29
MS13     22-8     41-1#
MS14     41-1#    50-46
MS15     41-1#    50-53
MS16     41-1#    47-44
MS17     41-1#    49-37
MS18     41-1#    61-43
MS19     41-1#    61-67
MS2      25-43    41-1#
MS20     41-1#    47-29
MS2000   40-38    41-1#
MS21     41-1#    47-38
MS22     41-1#    65-48
MS23     41-1#    65-71
MS24     34-70    41-1#    66-36
MS25     41-1#    66-55
MS26     41-1#    67-15
MS27     33-41    41-1#
MS28     34-96    41-1#
MS29     41-1#    59-28
MS3      25-47    41-1#
MS30     41-1#    60-33
MS31     41-1#    60-31
MS32     41-1#    60-35
```

CROSS REFERENCE TABLE (CREF V04.00)

```
OUT.06  24-11#  25-60*  25-65*  25-72*  29-26*  30-2*   33-41*  34-96*  47-31   60-31*  61-67*  65-71*  66-55*  67-15*
OUT.07  24-12#  25-61*  25-65*  25-72*  33-41*  34-96*  61-67*  65-71*  66-55*  67-15*
OUT.08  24-13#  25-66*  25-72*  33-44*  34-96*  61-67*  65-71*  66-55*  67-15*
OUT.09  24-14#  25-73*  34-96*  65-71*  66-55*  67-15*
OUT.10  24-15#  34-96*  65-71*  66-55*  67-15*
OUT.11  24-16#  34-96*  65-71*  66-55*  67-15*
OUT.12  24-17#  34-97*  65-72*  66-56*
OUT.13  24-18#
OUT.14  24-19#
OUT.15  24-20#
OUT.16  24-21#
OUT.17  24-22#
OUT.18  24-23#
OUT.19  24-24#  61-68*
OUT.20  24-25#
OUT.21  24-26#
OUT.22  24-27#
OUT.23  24-28#
OUT.24  24-29#
OUT.25  24-30#
OUT.26  24-31#
OUT.27  24-32#
OUT.28  24-33#
OUT.29  24-34#
OUT.30  24-35#
OUT.31  24-36#
OUT.32  24-37#
OUT.33  24-38#
OUT.34  24-39#
OUT.RQ  22-8*   23-14*  23-15   23-20   23-23*  24-5#   24-40   25-23*  25-43*  25-47*  25-51*  25-55*  25-59*  25-64*
        25-71*  29-26*  30-5    30-10   33-29*  33-33*  33-41*  34-60*  34-64*  34-70*  34-96*  40-38*  40-39   43-37*
        47-29*  47-44*  49-37*  50-46*  50-53*  54-19*  55-94*  58-111* 58-117* 58-119* 59-17*  59-20*  59-28*  60-31*
        60-33*  60-35*  61-35*  61-38*  61-43*  61-67*  65-38*  65-41*  65-48*  65-71*  66-26*  66-29*  66-36*  66-55*
        67-15*  67-16
OV...    3-14    3-14    3-14#  41-1    41-1    41-1    41-1#   42-2    42-2    42-2    42-2#   53-2    53-2    53-2
        53-2#   67-47   67-47#
OVE.MN   3-14#  41-1
OVE.MS  41-1#   42-2
OVE.SU  42-2#   53-2
OVE.TS  53-2#   67-47
OVERFL   6-75#
OVL...   3-14    3-14#  41-1    41-1#   42-2    42-2#   53-2    53-2#   67-47   67-47#
OVL.MN   3-14    3-14    41-1    41-1    41-1#
OVL.MS  27-28   41-1#   42-2    42-2    42-2#
OVL.SU  27-29   52-7#   53-2    53-2    53-2#
OVL.TS  27-30   67-38#  67-47   67-47   67-47#
OVRLAY  28-36   29-6#   52-5
OVRLNM  29-7    29-30*  29-34#  37-30*
OVRLPT  29-19   29-31   40-56#  42-2    42-2    53-2    53-2
OVRPNT  29-10*  29-26   29-35#
OVS.MN   3-14#
OVS.MS  41-1#
OVS.SU  42-2#
OVS.TS  53-2#
OVSTRT   4-18#  37-32   37-33
PAT4    36-3    36-10#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
PAT5     36-4     36-27#
PAT6     36-5     36-44#
PAT8     36-6     36-61#
PATPTR   36-2#    63-9     63-22
PHYSA     3-14     3-14     3-14     3-14     3-14#
PORT2    28-11#   28-52
PORT3    28-27#   28-31
PORT4    28-28    28-32#
PORT5    28-16    28-18    28-20    28-50#
RBNTRK    6-31#   49-5
RBUF0    34-16    34-53    34-87    34-96    34-96    34-96    36-85#
RBUFD    36-94    37-7#    58-59    58-113*  58-115*  66-18    67-8
RBUFLN    4-99#
RCONT     4-64#
RCTCPS    6-8#    44-19
RCTCSZ    6-40#
RCV       5-8#    25-34    37-100
RCVERR    7-25#   22-32    33-27    34-58    59-15    61-33    65-36    66-24
RCVRDY    7-21#   37-68    43-33
RDSTAT   22-15    22-20#   33-24    34-55    37-59    37-67    38-13    59-12    61-30    65-53    66-21
READ1    34-14#   50-95    51-61    51-69
READ1A   34-53#   34-95
READ1B   34-91    34-95#
READ1C   34-69    34-73#
READ1E   34-86    34-96#
READ1X   34-62    34-66    34-72    34-84    34-101#
READB    63-14    66-2#
REDERR   66-38    66-57#
REDEXT   66-35    66-52    66-54    66-61#
RETRY    35-60#   58-7*    58-102*  58-103*
RETS      6-7#
REVECT    6-63#
REVS      6-16#
RGO      66-33    66-39#
RM        6-32#
ROFDBN   35-80#   50-76*   50-77*
ROFDC    35-81#   50-48*   50-49*   50-60*   50-62*   50-68*   55-60*   55-62*   55-64*
RREAL     4-67#   36-96    58-58    66-16
RSTOP     4-63#   36-93    58-38    66-6
RTDS     22-5#    43-27    45-10    47-13
RTDSL    22-5     22-13#   58-24
RUN      35-29    35-45#   55-81*   56-5*    56-13*   56-15*   57-5*    57-8*    57-10*   57-14*   57-19*   57-22*   57-26*   57-27*
         57-29*   59-33*   60-4*    60-29*   60-33
RUNCMD   35-56#   56-15    59-33    60-29
RUNLBC   35-57#   57-27    60-4
RW.ANG    4-57#
RW.BUF    4-52#    4-53     58-60*   65-31*   66-19*
RW.CMD    4-55#    4-56     34-96    58-57*   58-58*   65-29*   66-17*
RW.HI     4-54#    4-55     34-96    58-56*   65-26*
RW.LOW    4-53#    4-54     34-16    34-87    34-96    58-47*   65-16*
RW.SDI    4-56#    4-57
RW.STA    4-48#    4-52     58-38*   58-83*   65-14*   66-14*
RWRDY     7-26#   33-37     34-68    47-40    58-29    61-41    65-44    66-32
SBCRES    7-44#
SCR      35-23    35-42#
SCTWRD    8-8#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
T4BB2     7-9#
T4C      49-36    49-39#
T4D      49-44    49-46#
T4MPRM    7-6#
T4MXFR    7-12#
T4SEEK    7-11#
T4SOFT    7-10#   51-74
T4UPRM    7-7#
T5       58-4#
T6       50-7#
T6A      50-11#   50-13
T6C      50-35#   50-39
T6D      50-36    50-38#
T6E      50-95#
T6F      50-88    50-90    50-103#
T6G      50-84#   50-98    50-101
T7       51-9#    67-38
T7A      51-15#   51-17
T7C      51-32#   51-51    51-88
T7X      51-55    52-1#
T8       55-2#
T8A      55-35#   55-49    55-68
T8B      55-42    55-44#
T8D      55-55    55-58    55-60#
T8E      55-70    55-75#
T8F      55-89    55-93    55-96#
T8T      55-65    55-69#
T9       59-2#
TALK     25-2#    30-14    33-17    44-7     44-33    45-6     45-27    46-7     46-20    47-6     47-23    48-14    55-10    55-91
         58-17    59-4     60-19    60-23
TEST     27-30    52-7     53-7#
TEST4     2-44#
TESTED   29-28    30-3     30-8#    58-120   60-32    60-34    60-36
TESTEV   30-4#    34-71    43-38    49-38    50-47    50-54    54-20    55-95    59-29    61-44
TESTEW   30-2#    33-30    33-34    34-61    34-65    43-31    45-14    47-15    47-45    59-18    59-21    61-36    61-39    65-39
         65-42    66-27    66-30
TESTEX   30-5#    33-20    33-45    34-98    44-10    44-36    45-9     45-30    46-10    46-23    47-10    48-17    55-13    59-7
         61-69    65-73    66-57
TESTEY   30-7#    47-32
TESTX    28-47#   29-32    55-17    59-35
TIMEOU    6-61#
TO       26-2#    44-13    44-17
TOOBIG    6-69#
TRACK    34-15    35-117#  58-57
TRKBOT   62-21    62-25#
TRKEXT   62-17    62-28#
TRKGRP    6-28#   50-8     51-10    55-46
TRKLOP   62-13#   62-27
TRKTST   55-40    62-2#
TSTBLK   31-54*   31-56*   35-88#   51-27*   51-28*   51-33    51-45*   51-46*   51-59    51-82*   51-84*   51-85*   51-87*   55-33*
         55-34*   55-41*   55-43*   61-7     61-15    62-7     62-9
TSTCM2   56-24    60-19#
TSTCMD   56-7     56-14    56-19    57-11    57-16    57-23    57-31    60-18#
TSTCME   60-25    60-37#
TSTCYL   31-22*   31-23*   31-24    31-63*   31-64*   31-65*   35-89#   48-28    48-29*   48-30*   48-31*   48-32*   50-87*   50-89*
         50-91    51-25*   51-26*   51-52*   51-54*   51-57    51-65    54-9*    54-10*   54-11*   54-12    55-27*   55-28*   55-29
```

```
                        ;ASSEMBLY CONTROL
                        ;
                        ;DEFINE THE SYMBOL ASS TO CONTROL THE ASSEMBLY OF THIS PROGRAM AS FOLLOWS:
100000                          FS=100000       ;FIELD SERVICE PROGRAM                                  (ZUDC)
040000                          EN=040000       ;ENGINEERING PROGRAM                                    (UDAHRD)
000001                          A1=000001       ;APT COMPATIBLE PROGRAM FOR FA&T, TEST 1-3              (IUDA)
000002                          A2=000002       ;APT COMPATIBLE PROGRAM FOR FA&T, TEST 4                (IUDB)
000004                          M1=000004       ;APT COMPATIBLE PROGRAM FOR MFG + EXTRAS, TEST 1-3      (IUDC)
000010                          M2=000010       ;APT COMPATIBLE PROGRAM FOR MFG + EXTRAS, TEST 4        (IUDD)

000002                  ASS=A2                  ;ASSEMBLY CONTROL DEFINITION

                        ;THE FOLLOWING ARE COMBINATIONS OF THE ABOVE DEFINITIONS FOR ASSEMBLY USE:
000005                          MD=A1+M1        ;APT COMPATIBLE, TEST 1-3
000012                          MX=A2+M2        ;APT COMPATIBLE, TEST 4
000017                          MM=MD+MX        ;ANY APT COMPATIBLE
000014                          MC=M1+M2        ;ANY APT COMPATIBLE FOR MFG + EXTRAS
140000                          EO=FS+EN        ;EXECUTABLE ONLY (READ DM PROGRAMS FROM A FILE)

```

```
                                .TITLE  TEST 4 QUESTIONS
                        .IF NZ FS$ASS
                                .ENABLE ABS
                                . = 2122
                        .IFF
                                .GLOBL T4QST
                        .ENDC
                                .MCALL  SVC
000000                          SVC
000000                          EQUALS
                        ;
                        ; BIT DIFINITIONS
                        ;
100000                  BIT15== 100000
040000                  BIT14== 40000
020000                  BIT13== 20000
010000                  BIT12== 10000
004000                  BIT11== 4000
002000                  BIT10== 2000
001000                  BIT09== 1000
000400                  BIT08== 400
000200                  BIT07== 200
000100                  BIT06== 100
000040                  BIT05== 40
000020                  BIT04== 20
000010                  BIT03== 10
000004                  BIT02== 4
000002                  BIT01== 2
000001                  BIT00== 1
                        ;
001000                  BIT9==  BIT09
000400                  BIT8==  BIT08
000200                  BIT7==  BIT07
000100                  BIT6==  BIT06
000040                  BIT5==  BIT05
000020                  BIT4==  BIT04
000010                  BIT3==  BIT03
000004                  BIT2==  BIT02
000002                  BIT1==  BIT01
000001                  BIT0==  BIT00
                        ;
                        ; EVENT FLAG DEFINITIONS
                        ;   EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION
                        ;
000040                  EF.START==      32.                     ; START COMMAND WAS ISSUED
000037                  EF.RESTART==    31.                     ; RESTART COMMAND WAS ISSUED
000036                  EF.CONTINUE==   30.                     ; CONTINUE COMMAND WAS ISSUED
000035                  EF.NEW==        29.                     ; A NEW PASS HAS BEEN STARTED
000034                  EF.PWR==        28.                     ; A POWER-FAIL/POWER-UP OCCURRED
                        ;
                        ;
                        ; PRIORITY LEVEL DEFINITIONS
                        ;
000340                  PRI07== 340
000300                  PRI06== 300
000240                  PRI05== 240
000200                  PRI04== 200
```

```
000140                  PRI03== 140
000100                  PRI02== 100
000040                  PRI01== 40
000000                  PRI00== 0
                        ;
                        ;OPERATOR FLAG BITS
                        ;
000004                  EVL==         4
000010                  LOT==        10
000020                  ADR==        20
000040                  IDU==        40
000100                  ISR==       100
000200                  UAM==       200
000400                  BOE==       400
001000                  PNT==      1000
002000                  PRI==      2000
004000                  IXE==      4000
010000                  IBE==     10000
020000                  IER==     20000
040000                  LOE==     40000
100000                  HOE==    100000
```

```
                        ;USEFUL INSTRUCTION DEFINITIONS
                        
                        .MACRO  AND ARG,ADR                     ;LOGICAL AND INSTRUCTION
                                .LIST
                                                                        BIC #^C<ARG>,ADR
                                .NLIST
                        .ENDM
                        
                        .MACRO  OR ARG,ADR                      ;LOGICAL OR INSTRUCTION
                                .LIST
                                                                        BIS #ARG,ADR
                                .NLIST
                        .ENDM
                        
                        .MACRO  PUSH ARG                        ;PUSH INSTRUCTION
                                .IRP X,<ARG>
                                .LIST
                                                                        MOV X,-(SP)
                                .NLIST
                                .ENDM
                        .ENDM
                        
                        .MACRO  POP ARG                         ;POP INSTRUCTION
                                .IRP X,<ARG>
                                .LIST
                                                                        MOV (SP)+,X
                                .NLIST
                                .ENDM
                        .ENDM
                        
                        .MACRO  .BR ADR                                 ;A BRANCH TO THE NEXT LOCATION
                                .IF P2
                                    .IF NE .-ADR
                                        .ERROR ;ILLEGAL .BR TO ADR
                                    .ENDC
                                .ENDC
                        .ENDM
                        
                        .MACRO  ASSUME FIRST CONDITION SECOND
                                    .IF CONDITION <FIRST>-<SECOND>
                                    .IFF
                                        .ERROR ;BAD ASSUME OF <FIRST> CONDITION <SECOND>
                                    .ENDC
                        .ENDM
```

```
                        ;MACRO DEFINITIONS FOR GLOBAL EQUATES

                        ;THESE MACROS ARE USED TO DEFINE INDEXES INTO A TABLE
                        ;
                        ;CALLING SEQUENCE MUST BE
                        ;
                        ;       TABLE
                        ;       ITEM    NAME    BYTES
                        ;       ITEM    NAME    BYTES
                        ;       ITEM    NAME    BYTES
                        ;       END     SIZE
                        ;
                        ;TABLE DEFINES THAT A TABLE IS ABOUT TO BE DEFINED AND END TERMINATES THE DEFINITION.
                        ;ANY NUMBER OF ITEM LINES CAN APPEAR. NAME IS THE NAME OF THE SYMBOL BEING EQUATED TO
                        ;THE INDEX. THE INDEX ALWAYS STARTS AT ZERO. BYTES SPECIFIES THE SIZE OF THE VALUE TO BE
                        ;STORED AT THAT INDEX IN BYTES. THE SIZE ARGUMENT TO THE END STATEMENT IS OPTIONAL, IT
                        ;BE EQUATED TO THE SIZE OF THE TABLE IN BYTES. THE SYMBOL TINDEX IS USED TO KEEP TRACK
                        ;OF THE INDEX VALUE AND WILL BE EQUAL TO THE SIZE OF THE TABLE AFTER THE END STATEMENT.

                        .MACRO TABLE
                                TINDEX=0
                        .ENDM

                        .MACRO ITEM NAME BYTES
                                NAME=TINDEX
                                TINDEX=TINDEX+BYTES
                        .ENDM

                        .MACRO END SIZE
                                .IF NB SIZE
                                SIZE=TINDEX
                                .ENDC
                        .ENDM
```

```
                                ;CONTROLLER TABLE DEFINITIONS
                                ;
                                ;ONE TABLE WILL BE SET UP BY INITIALIZE SECTION FOR EACH UDA SELECTED
                                ;FOR TESTING.  TABLES ARE CONTIGUOUS.  THE END OF THE TABLES IS
                                ;MARKED BY A WORD OF ZEROS.
                                ;
                                ;THE FIRST TABLE IS POINTED TO BY THE CONTENTS OF CTABS.
                                ;THE NUMBER OF TABLES IS CONTAINED IN CTRLRS.

000000                          TABLE                                   ;START A TABLE DEFINITION

000000                          ITEM C.UADR     2                       ;UNIBUS ADDRESS OF UDAIP REGISTER
000000                          ITEM C.UNIT     2
          000077                        CT.UNT= 000077                  ; LOGICAL UNIT NUMBER (FIRST)
          100000                        CT.AVL= BIT15                   ; SET WHEN NOT AVAILABLE FOR TESTING
000000                          ITEM C.VEC      2
          000777                        CT.VEC= 000777                  ; VECTOR ADDRESS
          007000                        CT.BRL= 007000                  ; BR LEVEL
000000                          ITEM C.BST      2                       ; BURST LEVEL
000000                          ITEM C.JSR      2                       ;INTERRUPT SERVICE ROUTINE FOR CONTROLLER
000000                          ITEM C.JAD      2                       ;  THESE TWO WORDS LOADED WITH [JSR R0,UDASRV]
000000                          ITEM C.FLG      2                       ;FLAGS
          000002                        CT.RN=  BIT1                    ;DM PROGRAM RUNNING
          000004                        CT.CMD= BIT2                    ;COMMAND ISSUED, WAITING FOR RESPONSE
          000010                        CT.MSG= BIT3                    ;MESSAGE RESPONSE RECEIVED
                                                                        ;WHENEVER THIS BIT IS SET, CT.CMD IS CLEARED
          000020                        CT.REQ= BIT4                    ;BUFFER HAS BEEN GIVEN TO UDA FOR REQUEST
                                                                        ;SET WHENEVER READ STUD DATA COMMAND
                                                                        ;GIVEN TO UDA
000000                          ITEM C.RING     2                       ;RING BUFFER ADDRESS
000000                          ITEM C.DR0      2                       ;POINTER TO DRIVE TABLES
000000                          ITEM C.DR1      2                       ;  IF ZERO, NO DRIVE TABLE EXISTS
000000                          ITEM C.DR2      2
000000                          ITEM C.DR3      2
000000                          ITEM C.DR4      2
000000                          ITEM C.DR5      2
000000                          ITEM C.DR6      2
000000                          ITEM C.DR7      2
000000                          ITEM C.TO       2                       ;TIMEOUT COUNTER
000000                          ITEM C.TOH      2                       ;  (TWO WORDS)
000000                          ITEM C.REF      2                       ;COMMAND REFERENCE NUMBER

000000                          END C.SIZE                              ;SIZE OF CONTROLLER TABLE IN BYTES
```

```
                                ;DRIVE TABLE DEFINITIONS
                                ;
                                ;ONE DRIVE TABLE WILL BE SET UP BY THE INITIALIZE SECTION FOR EACH
                                ;DRIVE SELECTED FOR TESTING.  EACH TABLE IS POINTED TO BY A
                                ;WORD IN THE CONTROLLER TABLE ON WHICH THE DRIVE EXISTS.

000000                          TABLE                                   ;START A TABLE DEFINITION

000000                          ITEM D.DRV      2                       ;DRIVE NUMBER
000000                          ITEM D.UNIT     2
                                        DT.UNT= 000077                  ; LOGICAL UNIT NUMBER OF DRIVE
                                        DT.AVL= BIT15                   ; SET WHEN NOT AVAILABLE FOR TESTING
000000                          ITEM D.PRM      2                       ;HARDWARE QUESTION FLAGS
                                        D.IW    =BIT14                  ;INITIAL WRITE
                                        D.DCY   =BIT13                  ;DIAGNOSTIC CYLINDERS
                                        D.ECC   =BIT12                  ;ECC CORRECTION ENABLED
                                        D.RO    =BIT11                  ;READ ONLY
                                        D.WO    =BIT10                  ;WRITE ONLY
                                        D.RET   =BIT9                   ;RETRIES ENABLED
                                        D.CYL   =BIT8                   ;START/END CYLINDERS SPECIFIED
                                        D.SEQ   =BIT6                   ;SEQUENTIAL ACCESS
                                        D.BE    =BIT5                   ;BEGIN/END BLOCKS USED
                                        D.TR    =BIT4                   ;WHEN D.BE=0: 1 - TRACKS, 0 - GROUPS
                                        D.WC    =BIT3                   ;WRITE CHECKS ENABLED
                                        D.WCA   =BIT2                   ;ALWAYS WRITE CHECK
                                        D.DC    =BIT1                   ;DATA COMPARES ENABLED
                                        D.DCA   =BIT0                   ;ALWAYS DATA COMPARE
                                        DDEF=D.ECC+D.WC+D.DC+D.RET      ;DEFAULT D.PRM
                                        D.ZERO=BIT15+BIT7+D.IW          ;BITS TO BE CLEARED
000000                          ITEM D.PAT      2                       ;DATA PATTERN NUMBER
000000                          ITEM D.BB       2                       ;BAD BLOCK COUNT
000000                          ITEM D.BB01     4                       ;BAD BLOCK 1
000000                          ITEM D.BB02     4                       ;       2
000000                          ITEM D.BB03     4                       ;       3
000000                          ITEM D.BB04     4                       ;       4
000000                          ITEM D.BB05     4                       ;       5
000000                          ITEM D.BB06     4                       ;       6
000000                          ITEM D.BB07     4                       ;       7
000000                          ITEM D.BB08     4                       ;       8
000000                          ITEM D.BB09     4                       ;       9
000000                          ITEM D.BB10     4                       ;      10
000000                          ITEM D.BB11     4                       ;      11
000000                          ITEM D.BB12     4                       ;      12
000000                          ITEM D.BB13     4                       ;      13
000000                          ITEM D.BB14     4                       ;      14
000000                          ITEM D.BB15     4                       ;      15
000000                          ITEM D.BB16     4                       ;      16
```

```
 1 000000                ITEM D.BEC      2       ;BEGIN/END SET COUNT
 2 000000                ITEM D.BGN1     4       ;BEGIN BLOCK 1
 3 000000                ITEM D.END1     4       ;END
 4 000000                ITEM D.BGN2     4       ;BEGIN BLOCK 2
 5 000000                ITEM D.END2     4       ;END
 6 000000                ITEM D.BGN3     4       ;BEGIN BLOCK 3
 7 000000                ITEM D.END3     4       ;END
 8 000C00                ITEM D.BGN4     4       ;BEGIN BLOCK 4
 9 000000                ITEM D.END4     4       ;END
10 000000                ITEM D.BCYL     4       ;BEGIN CYLINDER
11 000000                ITEM D.ECYL     4       ;END CYLINDER
12 000000                ITEM D.XFRW     2       ;MEGABITS WRITTEN COUNT
13 000000                ITEM D.XFRR     2       ;MEGABITS READ COUNT
14 000C00                ITEM D.HERR     2       ;HARD ERROR COUNTER
15 000000                ITEM D.SERR     2       ;SOFT ERROR COUNTER
16 000000                ITEM D.SEEK     2       ;NUMBER OF SEEKS X1000
17 000000                ITEM D.ECCC     2       ;ECC COUNTER
18 000000                ITEM D.SERN     6       ;DRIVE SERIAL NUMBER
27 000000                END D.SIZE              ;SIZE OF DRIVE TABLE IN BYTES
28
```

```
                                        .LIST MEB
000000                          T4QST::
000000  000402                        BR      T4QUES          ; BRANCH TO TEST 4 QUESTIONS
                                ;                       INPUT - NONE
000002  000167  001774                  JMP     T4QUE2          ; BRANCH TO LAST OF THE TEST 4 QUESTIONS
                                ;                       INPUT - R1 -> PATTERN 16 SIZE
```

```
                                ;ASK TEST 4 MANUAL INTERVENTION QUESTIONS
                                ;
                                ;INPUTS:
                                ;       R5 - POINTER TO CONTROLLER TABLE
                                ;       R3 - POINTER TO DRIVE TABLE
                                ;       R2 AND R4 MUST BE PRESERVED
                                ;OUTPUTS:
                                ;       DRIVE TABLE WITH NEW PARAMETERS
                                ;       R0 AND R1 CONTENTS DESTROYED
                                
000006                          T4QUEST:PUSH <R2,R4>
000006  010246                                                                 MOV R2,-(SP)
000010  010446                                                                 MOV R4,-(SP)
000012                                  PRINTF #T4QHED,D.UNIT(R3),(R5),(R3)     ;PRINT HEADER
000012  011346                  MOV     (R3),-(SP)
000014  011546                  MOV     (R5),-(SP)
000016  016346  000002          MOV     D.UNIT(R3),-(SP)
000022  012746  003506'          MOV     #T4QHED,-(SP)
000026  012746  000004          MOV     #4,-(SP)
000032  010600                  MOV     SP,R0
000034  104417                  TRAP    C$PNTF
000036  062706  000012          ADD     #12,SP
000042  016367  000010  004246          MOV D.BB(R3),TEMP
000050                                  GMANID T4BB,TEMP,D,-1,0,16.,YES ;NUMBER OF BAD BLOCKS
000050  104443                  TRAP    C$GMAN
000052  000406                  BR      10000$
000054  004316'                 .WORD   TEMP
000056  000052                  .WORD   T$CODE
000060  002440'                 .WORD   T4BB
000062  177777                  .WORD   -1
000064  000000                  .WORD   T$LOLIM
000066  000020                  .WORD   T$HILIM
000070  016763  004222  000010          MOV TEMP,D.BB(R3)
000076  001424                          BEQ T4Q02
                                
000100  010304                          MOV R3,R4                       ;GET POINTER TO STORAGE
000102  062704  000012                  ADD #D.BB01,R4                  ;  FOR BAD BLOCKS
000106  016701  004204                  MOV TEMP,R1                     ;GET COUNT OF BLOCKS TO INPUT
```

```
 1 000112  004767  002030          T4Q01:  CALL BLD28                          ;BUILD DEFAULT ANSWER
 2 000116                                  GMANID T4BBI,TEMP,A,-1,0,9.,YES ;BAD BLOCK
   000116  104443                  TRAP    C$GMAN
   000120  000406                  BR      10001$
   000122  004316'                 .WORD   TEMP
   000124  000152                  .WORD   T$CODE
   000126  002555'                 .WORD   T4BBI
   000130  177777                  .WORD   -1
   000132  000000                  .WORD   T$LOLIM
   000134  000011                  .WORD   T$HILIM
 3 000136  004767  002106                  CALL CNV28                          ;CONVERT TO BINARY
 4 000142  103763                          BCS T4Q01                           ;REPEAT UNTIL RIGHT
 5 000144  005301                          DEC R1                              ;DECREMENT COUNT
 6 000146  001361                          BNE T4Q01                           ;GET ALL NUMBERS
 7 000150                          T4Q02:
 8 000150  005067  004166                  CLR     ANYMOR                      ; DEFAULT ANYMORE WITH 0
 9 000154                                  GMANIL  T4DMN,ANYMOR,ANY.YS,YES
   000154  104443                  TRAP    C$GMAN
   000156  000404                  BR      10002$
   000160  004342'                 .WORD   ANYMOR
   000162  000130                  .WORD   T$CODE
   000164  002465'                 .WORD   T4DMN
   000166  000001                  .WORD   ANY.YS
10 000170  032767  000001  004144          BIT     #ANY.YS,ANYMOR              ;DO WE ASK ANY MORE QUESTIONS?
11 000176  001002                          BNE     1$                          ;  IF SO, CONTINUE
12 000200  000167  001570                  JMP     T4Q30                       ; ELSE EXIT
13 000204                          1$:
14 000204  016367  000004  004104          MOV D.PRM(R3),TEMP                  ;GET PARAMETER BITS
15 000212                                  GMANIL T4RO,TEMP,D.RO,YES           ;READ ONLY
   000212  104443                  TRAP    C$GMAN
   000214  000404                  BR      10003$
   000216  004316'                 .WORD   TEMP
   000220  000130                  .WORD   T$CODE
   000222  002567'                 .WORD   T4RO
   000224  004000                  .WORD   D.RO
16 000226  032767  004000  004062          BIT #D.RO,TEMP
17 000234  001404                          BEQ T4Q03                           ;IF NOT READ ONLY, GO TO WRITE ONLY QUESTION
18 000236  042767  002000  004052          BIC #D.WO,TEMP                      ;ELSE, CLEAR WRITE ONLY BIT
19 000244  000432                          BR T4Q05                            ; AND BRANCH AROUND WRITE ONLY QUESTION
20 000246                          T4Q03:
21 000246                                  GMANIL T4WO,TEMP,D.WO,YES           ;WRITE ONLY
   000246  104443                  TRAP    C$GMAN
   000250  000404                  BR      10004$
   000252  004316'                 .WORD   TEMP
   000254  000130                  .WORD   T$CODE
   000256  002601'                 .WORD   T4WO
   000260  002000                  .WORD   D.WO
22 000262                                  GMANIL T4WCA,TEMP,D.WCA,YES         ;CHECK ALL WRITES
   000262  104443                  TRAP    C$GMAN
   000264  000404                  BR      10005$
   000266  004316'                 .WORD   TEMP
   000270  000130                  .WORD   T$CODE
   000272  002614'                 .WORD   T4WCA
   000274  000004                  .WORD   D.WCA
23 000276  032767  000004  004012          BIT #D.WCA,TEMP                     ;CHECK ANSWER
24 000304  001007                          BNE T4Q04                           ;BRANCH IF YES
25 000306                                  GMANIL T4WCR,TEMP,D.WC,YES          ;RANDOMLY CHECK WRITES
```

```
000306  104443                  TRAP    C$GMAN
000310  000404                  BR      10006$
000312  004316'                 .WORD   TEMP
000314  000130                  .WORD   T$CODE
000316  002650'                 .WORD   T4WCR
000320  000010                  .WORD   D.WC
     26 000322  000403                  BR T4Q05
```

```
 1 000324  052767  000010  003764  T4Q04:  BIS #D.WC,TEMP                      ;BOTH BITS GET SET
 2 000332  016763  003760  000004  T4Q05:  MOV TEMP,D.PRM(R3)                  ;PUT PARAM BITS BACK
 3 000340  016367  000006  003750          MOV D.PAT(R3),TEMP
 4 000346                                  GMANID T4DP,TEMP,D,-1,0,16.,YES ;DATA PATTERN
   000346  104443                  TRAP    C$GMAN
   000350  000406                  BR      10007$
   000352  004316'                 .WORD   TEMP
   000354  000052                  .WORD   T$CODE
   000356  002711'                 .WORD   T4DP
   000360  177777                  .WORD   -1
   000362  000000                  .WORD   T$LOLIM
   000364  000020                  .WORD   T$HILIM
 5 000366  016763  003724  000006          MOV TEMP,D.PAT(R3)
 6 000374  016367  000004  003714          MOV D.PRM(R3),TEMP                  ;GET PARAM BITS AGAIN
 7 000402  032767  004000  003706  T4Q06:  BIT #D.RO,TEMP                     ;BYPASS NEXT 3 IF ONLY WRITING
 8 000410  001010                          BNE T4Q07
 9 000412  032767  002000  003676          BIT #D.WO,TEMP
10 000420  001404                          BEQ T4Q07
11 000422  032767  000010  003666          BIT #D.WC,TEMP
12 000430  001432                          BEQ T4Q09
13 000432                          T4Q07:  GMANIL T4ECC,TEMP,D.ECC,YES         ;ENABLE ECC
   000432  104443                  TRAP    C$GMAN
   000434  000404                  BR      10010$
   000436  004316'                 .WORD   TEMP
   000440  000130                  .WORD   T$CODE
   000442  002757'                 .WORD   T4ECC
   000444  010000                  .WORD   D.ECC
14 000446                                  GMANIL T4DCA,TEMP,D.DCA,YES         ;COMPARE ALL DATA
   000446  104443                  TRAP    C$GMAN
   000450  000404                  BR      10011$
   000452  004316'                 .WORD   TEMP
   000454  000130                  .WORD   T$CODE
   000456  003012'                 .WORD   T4DCA
   000460  000001                  .WORD   D.DCA
15 000462  032767  000001  003626          BIT #D.DCA,TEMP                     ;CHECK ANSWER
16 000470  001007                          BNE T4Q08                           ;BRANCH IF YES
17 000472                                  GMANIL T4DCR,TEMP,D.DC,YES          ;RANDOMLY CHECK WRITES
   000472  104443                  TRAP    C$GMAN
   000474  000404                  BR      10012$
   000476  004316'                 .WORD   TEMP
   000500  000130                  .WORD   T$CODE
   000502  003040'                 .WORD   T4DCR
   000504  000002                  .WORD   D.DC
18 000506  000403                          BR T4Q09
```

```
 1 000510  052767  000002  003600  T4Q08:  BIS #D.DC,TEMP                  ;BOTH BITS GET SET
 2 000516                          T4Q09:  GMANIL T4RET,TEMP,D.RET,YES     ;ENABLE RETRIES
   000516  104443                  TRAP    C$GMAN
   000520  000404                  BR      10013$
   000522  004316'                 .WORD   TEMP
   000524  000130                  .WORD   T$CODE
   000526  003073'                 .WORD   T4RET
   000530  001000                  .WORD   D.RET
 3 000532  005167  003560                  COM TEMP
 4 000536                                  GMANIL T4SEK,TEMP,D.SEQ,YES     ;ENABLE SEEKS
   000536  104443                  TRAP    C$GMAN
   000540  000404                  BR      10014$
   000542  004316'                 .WORD   TEMP
   000544  000130                  .WORD   T$CODE
   000546  003112'                 .WORD   T4SEK
   000550  000100                  .WORD   D.SEQ
 5 000552  005167  003540                  COM TEMP                        ;COMPLIMENTED
 6 000556  016763  003534  000004          MOV TEMP,D.PRM(R3)
 7
 8 000564  005067  003526                  CLR TEMP                        ;DETERMINE DEFAULT SELECTION
 9 000570  032763  000040  000004          BIT #D.BE,D.PRM(R3)             ;IF D.BE SET - LOAD 1
10 000576  001403                          BEQ T4Q10                       ;IF D.CYL CLEAR - LOAD 0
11 000600  005267  003512                  INC TEMP                        ;IF D.BEC CONTAINS 0 - LOAD 4
12 000604  000422                          BR T4Q11                        ;IF D.TR SET - LOAD 2
13 000606  032763  000400  000004  T4Q10:  BIT #D.CYL,D.PRM(R3)            ;LOAD 3
14 000614  001416                          BEQ T4Q11
15 000616  012767  000004  003472          MOV #4,TEMP
16 000624  005763  000112                  TST D.BEC(R3)
17 000630  001410                          BEQ T4Q11
18 000632  005367  003460                  DEC TEMP
19 000636  032763  000020  000004          BIT #D.TR,D.PRM(R3)
20 000644  001402                          BEQ T4Q11
21 000646  005367  003444                  DEC TEMP
```

```
     1 000652                                T4Q11:  PRINTF #T4OPT1
       000652  012746  003617'                MOV     #T4OPT1,-(SP)
       000656  012746  000001                MOV     #1,-(SP)
       000662  010600                         MOV     SP,R0
       000664  104417                         TRAP    C$PNTF
       000666  062706  000004                 ADD     #4,SP
     2 000672                                        PRINTF #T4OPT2
       000672  012746  003645'                MOV     #T4OPT2,-(SP)
       000676  012746  000001                MOV     #1,-(SP)
       000702  010600                         MOV     SP,R0
       000704  104417                         TRAP    C$PNTF
       000706  062706  000004                 ADD     #4,SP
     3 000712                                        PRINTF #T4OPT3
       000712  012746  003712'                MOV     #T4OPT3,-(SP)
       000716  012746  000001                MOV     #1,-(SP)
       000722  010600                         MOV     SP,R0
       000724  104417                         TRAP    C$PNTF
       000726  062706  000004                 ADD     #4,SP
     4 000732                                        PRINTF #T4OPT4
       000732  012746  003764'                MOV     #T4OPT4,-(SP)
       000736  012746  000001                MOV     #1,-(SP)
       000742  010600                         MOV     SP,R0
       000744  104417                         TRAP    C$PNTF
       000746  062706  000004                 ADD     #4,SP
     5 000752                                        PRINTF #T4OPT5
       000752  012746  004044'                MOV     #T4OPT5,-(SP)
       000756  012746  000001                MOV     #1,-(SP)
       000762  010600                         MOV     SP,R0
       000764  104417                         TRAP    C$PNTF
       000766  062706  000004                 ADD     #4,SP
     6 000772                                        PRINTF #T4OPT6
       000772  012746  004124'                MOV     #T4OPT6,-(SP)
       000776  012746  000001                MOV     #1,-(SP)
       001002  010600                         MOV     SP,R0
       001004  104417                         TRAP    C$PNTF
       001006  062706  000004                 ADD     #4,SP
     7 001012                                        GMANID T4OPT7,TEMP,D,-1,0,4,YES ;WHICH SELECTION LIMITS
       001012  104443                         TRAP    C$GMAN
       001014  000406                         BR      10015$
       001016  004316'                       .WORD   TEMP
       001020  000052                       .WORD   T$CODE
       001022  003135'                       .WORD   T4OPT7
       001024  177777                       .WORD   -1
       001026  000000                       .WORD   T$LOLIM
       001030  000004                       .WORD   T$HILIM
     8 001032  005367  003260                        DEC TEMP                        ;SET UP D.PRM FROM ANSWER
     9 001036  002004                               BGE T4Q12                       ;IF 0 - CLEAR D.BE AND D.CYL
    10 001040  042763  000440  000004               BIC #D.BE+D.CYL,D.PRM(R3)
    11 001046  000467                               BR T4Q19
```

```
 1 001050  005367  003242          T4Q12:  DEC TEMP                    ;IF 1
 2 001054  002013                          BGE T4Q13                   ;  IF D.BE NOT SET
 3 001056  032763  000040  000004          BIT #D.BE,D.PRM(R3)         ;    SET D.BE
 4 001064  001060                          BNE T4Q19                   ;    CLEAR D.CYL
 5 001066  052763  000040  000004          BIS #D.BE,D.PRM(R3)         ;    LOAD 1 IN D.BEC
 6 001074  042763  000400  000004          BIC #D.CYL,D.PRM(R3)        ;    CLEAR BLOCK STORAGE
 7 001102  000436                          BR T4Q16
 8 001104  042763  000040  000004  T4Q13:  BIC #D.BE,D.PRM(R3)        ;IF 2, 3 OR 4
 9                                                                    ;  CLEAR D.BE
10 001112  022767  000002  003176          CMP #2,TEMP                 ;IF 4
11 001120  001006                          BNE T4Q14                   ;  SET D.CYL
12 001122  052763  000400  000004          BIS #D.CYL,D.PRM(R3)        ;  CLEAR D.BEC
13 001130  005063  000112                  CLR D.BEC(R3)
14 001134  000434                          BR T4Q19
15 001136                          T4Q14:  PUSH D.PRM(R3)              ;IF 2 OR 3
   001136  016346  000004                                                     MOV D.PRM(R3),-(SP)
16 001142  052763  000420  000004          BIS #D.CYL+D.TR,D.PRM(R3)    ;  SAVE D.PRM BITS
17 001150  005367  003142                  DEC TEMP                    ;  SET D.CYL AND D.TR
18 001154  100403                          BMI T4Q15                   ;  IF 3
19 001156  042763  000020  000004          BIC #D.TR,D.PRM(R3)          ;    CLEAR D.TR
20 001164  022663  000004          T4Q15:  CMP (SP)+,D.PRM(R3)         ;  IF D.CYL OR D.TR CHANGED OR D.BEC = 0
21 001170  001003                          BNE T4Q16
22 001172  005763  000112                  TST D.BEC(R3)               ;    LOAD 1 IN D.BEC
23 001176  001013                          BNE T4Q19                   ;    CLEAR BLOCK STORAGE
24 001200  012763  000001  000112  T4Q16:  MOV #1,D.BEC(R3)
25 001206  010304                  T4Q17:  MOV R3,R4
26 001210  062704  000114                  ADD #D.BGN1,R4
27 001214  012701  000020                  MOV #16.,R1
28 001220  005024                  T4Q18:  CLR (R4)+
29 001222  005301                          DEC R1
30 001224  001375                          BNE T4Q18
```

```
 1 001226  032763  000040  000004  T4Q19:  BIT #D.BE,D.PRM(R3)              ;NOW ASK THE QUESTIONS TO ALLOW THE
 2 001234  001460                          BEQ T4Q22                        ;  NUMBERS TO CHANGE
 3 001236  016367  000112  003052          MOV D.BEC(R3),TEMP
 4 001244                                  GMANID T4BE,TEMP,D,-1,1,4,YES    ;NUMBER OF B/E SETS
   001244  104443                  TRAP    C$GMAN
   001246  000406                  BR      10016$
   001250  004316'                 .WORD   TEMP
   001252  000052                  .WORD   T$CODE
   001254  003140'                 .WORD   T4BE
   001256  177777                  .WORD   -1
   001260  000001                  .WORD   T$LOLIM
   001262  000004                  .WORD   T$HILIM
 5 001264  016763  003026  000112          MOV TEMP,D.BEC(R3)
 6 001272  016701  003020                  MOV TEMP,R1                      ;GET COUNT OF SETS
 7 001276  010304                          MOV R3,R4                        ;GET POINTER TO STORAGE AREA
 8 001300  062704  000114                  ADD #D.BGN1,R4
 9 001304  004767  000636          T4Q20:  CALL BLD28
10 001310                                  GMANID T4BEG,TEMP,A,-1,0,9.,YES ;BEGIN BLOCK
   001310  104443                  TRAP    C$GMAN
   001312  000406                  BR      10017$
   001314  004316'                 .WORD   TEMP
   001316  000152                  .WORD   T$CODE
   001320  003171'                 .WORD   T4BEG
   001322  177777                  .WORD   -1
   001324  000000                  .WORD   T$LOLIM
   001326  000011                  .WORD   T$HILIM
11 001330  004767  000714                  CALL CNV28
12 001334  103763                          BCS T4Q20
13 001336  004767  000604          T4Q21:  CALL BLD28
14 001342                                  GMANID T4END,TEMP,A,-1,0,9.,YES ;END BLOCK
   001342  104443                  TRAP    C$GMAN
   001344  000406                  BR      10020$
   001346  004316'                 .WORD   TEMP
   001350  000152                  .WORD   T$CODE
   001352  003205'                 .WORD   T4END
   001354  177777                  .WORD   -1
   001356  000000                  .WORD   T$LOLIM
   001360  000011                  .WORD   T$HILIM
15 001362  004767  000662                  CALL CNV28
16 001366  103763                          BCS T4Q21
17 001370  005301                          DEC R1
18 001372  001344                          BNE T4Q20
19 001374  000577                          BR T4Q30
```

```
 1 001376  032763  000400  000004  T4Q22:  BIT #D.CYL,D.PRM(R3)            ;IF D.CYL CLEAR - ALL DONE
 2 001404  001573                          BEQ T4Q30
 3 001406  005763  000112                  TST D.BEC(R3)                  ;IF D.BEC CLEAR - GO RIGHT TO B/F CYLS
 4 001412  001526                          BEQ T4Q27
 5 001414  010304                          MOV R3,R4
 6 001416  062704  000112                  ADD #D.BEC,R4
 7 001422  032763  000020  000004          BIT #D.TR,D.PRM(R3)            ;LOOK AT D.TR.TO DETERMINE QUESTION
 8 001430  001434                          BEQ T4Q24
 9 001432  011467  002660                  MOV (R4),TEMP
10 001436                                  GMANID T4TRC,TEMP,D,-1,1,7,YES ;NUMBER OF TRACKS
   001436  104443                TRAP      C$GMAN
   001440  000406                BR        10021$
   001442  004316'               .WORD     TEMP
   001444  000052               .WORD     T$CODE
   001446  003217'               .WORD     T4TRC
   001450  177777               .WORD     -1
   001452  000001               .WORD     T$LOLIM
   001454  000007               .WORD     T$HILIM
11 001456  016714  002634                  MOV TEMP,(R4)
12 001462  012401                          MOV (R4)+,R1                   ;GET COUNT OF TRACKS
13 001464  011467  002626          T4Q23:  MOV (R4),TEMP
14 001470                                  GMANID T4TRAK,TEMP,D,-1,0,255.,YES      ;TRACK
   001470  104443                TRAP      C$GMAN
   001472  000406                BR        10022$
   001474  004316'               .WORD     TEMP
   001476  000052               .WORD     T$CODE
   001500  003250'               .WORD     T4TRAK
   001502  177777               .WORD     -1
   001504  000000               .WORD     T$LOLIM
   001506  000377               .WORD     T$HILIM
15 001510  016724  002602                  MOV TEMP,(R4)+
16 001514  005301                          DEC R1
17 001516  001362                          BNE T4Q23
18 001520  000433                          BR T4Q26
19 001522  011467  002570          T4Q24:  MOV (R4),TEMP
20 001526                                  GMANID T4GRC,TEMP,D,-1,1,7,YES  ;NUMBER OF GROUPS
   001526  104443                TRAP      C$GMAN
   001530  000406                BR        10023$
   001532  004316'               .WORD     TEMP
   001534  000052               .WORD     T$CODE
   001536  003256'               .WORD     T4GRC
   001540  177777               .WORD     -1
   001542  000001               .WORD     T$LOLIM
   001544  000007               .WORD     T$HILIM
21 001546  016714  002544                  MOV TEMP,(R4)
22 001552  012401                          MOV (R4)+,R1                   ;GET COUNT OF GROUPS
```

```
     1 001554  011467  002536          T4Q25:  MOV (R4),TEMP
     2 001560                                  GMANID T4GRP,TEMP,D,-1,0,255.,YES       ;GROUP
       001560  104443                  TRAP    C$GMAN
       001562  000406                  BR      10024$
       001564  004316'                 .WORD   TEMP
       001566  000052                  .WORD   T$CODE
       001570  003307'                 .WORD   T4GRP
       001572  177777                  .WORD   -1
       001574  000000                  .WORD   T$LOLIM
       001576  000377                  .WORD   T$HILIM
     3 001600  016724  002512                  MOV TEMP,(R4)+
     4 001604  005301                          DEC R1
     5 001606  001362                          BNE T4Q25
     6 001610  016367  000162  002500  T4Q26:  MOV D.ECYL+2(R3),TEMP
     7 001616  005167  002474                  COM TEMP
     8 001622                                  GMANIL T4CYL,TEMP,BIT15,YES     ;WISH TO LIMIT CYLINDERS
       001622  104443                  TRAP    C$GMAN
       001624  000404                  BR      10025$
       001626  004316'                 .WORD   TEMP
       001630  000130                  .WORD   T$CODE
       001632  003315'                 .WORD   T4CYL
       001634  100000                  .WORD   BIT15
     9 001636  005767  002454                  TST TEMP
    10 001642  100412                          BMI T4Q27
    11 001644  005063  000154                  CLR D.BCYL(R3)
    12 001650  005063  000156                  CLR D.BCYL+2(R3)
    13 001654  005063  000160                  CLR D.ECYL(R3)
    14 001660  012763  177777  000162          MOV #-1,D.ECYL+2(R3)
    15 001666  000442                          BR T4Q30
    16 001670  005763  000162          T4Q27:  TST D.ECYL+2(R3)
    17 001674  002002                          BGE T4Q27A
    18 001676  005063  000162                  CLR D.ECYL+2(R3)
    19 001702  010304                  T4Q27A: MOV R3,R4
    20 001704  062704  000154                  ADD #D.BCYL,R4
    21 001710  004767  000232          T4Q28:  CALL BLD28
    22 001714                                  GMANID T4CYLB,TEMP,A,-1,0,9.,YES        ;STARTING CYLINDER
       001714  104443                  TRAP    C$GMAN
       001716  000406                  BR      10026$
       001720  004316'                 .WORD   TEMP
       001722  000152                  .WORD   T$CODE
       001724  003367'                 .WORD   T4CYLB
       001726  177777                  .WORD   -1
       001730  000000                  .WORD   T$LOLIM
       001732  000011                  .WORD   T$HILIM
    23 001734  004767  000310                  CALL CNV28
    24 001740  103763                          BCS T4Q28
```

```
      1 001742  004767  000200          T4Q29:  CALL BLD28
      2 001746                                  GMANID T4CYLE,TEMP,A,-1,0,9.,YES        ;ENDING CYLINDER
        001746  104443                    TRAP    C$GMAN
        001750  000406                    BR      10027$
        001752  004316'                   .WORD   TEMP
        001754  000152                    .WORD   T$CODE
        001756  003411'                   .WORD   T4CYLE
        001760  177777                    .WORD   -1
        001762  000000                    .WORD   T$LOLIM
        001764  000011                    .WORD   T$HILIM
      3 001766  004767  000256                    CALL CNV28
      4 001772  103763                          BCS T4Q29
      5 001774                          T4Q30:  POP <R4,R2>
        001774  012604                                                          MOV (SP)+,R4
        001776  012602                                                          MOV (SP)+,R2
      6 002000  000207                          RETURN
```

```
 1                                      ;NOW GET DATA PATTERN 16 IF SELECTED BY ANY DRIVE
 2                                      ;
 3                                      ;       INPUT R1 POINTS TO PATTERN 16 SIZE
 4
 5 002002                               T4QUE2:
 6 002002  032767  000001  002332              BIT     #ANY.YS,ANYMOR          ; DO WE ASK THESE QUESTIONS?
 7 002010  001436                              BEQ     T4QU2E                  ; IF NOT, EXIT
 8 002012  011167  002300                      MOV     (R1),TEMP
 9 002016                                      GMANID T4DPC,TEMP,D,-1,0,16.,YES        ;COUNT OF WORDS
   002016  104443                      TRAP    C$GMAN
   002020  000406                      BR      10030$
   002022  004316'                     .WORD   TEMP
   002024  000052                     .WORD   T$CODE
   002026  003431'                     .WORD   T4DPC
   002030  177777                     .WORD   -1
   002032  000000                     .WORD   T$LOLIM
   002034  000020                     .WORD   T$HILIM
10 002036  016721  002254                      MOV     TEMP,(R1)+              ; GET COUNT OF WORDS
11 002042  010104                              MOV     R1,R4                   ; R4 HAS ADDRESS OF STORAGE
12 002044  016701  002246                      MOV     TEMP,R1
13 002050  001416                              BEQ     T4QU2E                  ; IF 0, NO DATA WORDS
14 002052  011467  002240            T4PRM5:  MOV (R4),TEMP
15 002056                                      GMANID T4DPD,TEMP,0,-1,0,-1,YES ;DATA WORD
   002056  104443                      TRAP    C$GMAN
   002060  000406                      BR      10031$
   002062  004316'                     .WORD   TEMP
   002064  000032                     .WORD   T$CODE
   002066  003474'                     .WORD   T4DPD
   002070  177777                     .WORD   -1
   002072  000000                     .WORD   T$LOLIM
   002074  177777                     .WORD   T$HILIM
16 002076  016724  002214                      MOV TEMP,(R4)+
17 002102  005301                              DEC R1                                  ;COUNT THE WORDS
18 002104  001362                              BNE T4PRM5
19 002106  000207                    T4QU2E:   RETURN
```

```
                                ;DIVIDE
                                ;
                                ;DIVIDE A 32 BIT UNSIGNED NUMBER BY A 16 BIT UNSIGNED NUMBER.
                                ;REPLACE DIVIDEND WITH QUOTIENT AND RETURN REMAINDER.
                                ;WILL NOT CHECK FOR DIVIDE BY ZERO.
                                ;
                                ;INPUTS:
                                ;       R3 - LOW 16 BITS OF DIVIDEND
                                ;       R4 - HIGH 16 BITS OF DIVIDEND
                                ;       R0 - DIVISOR
                                ;OUTPUTS:
                                ;       R3 - LOW 16 BITS OF QUOTIENT
                                ;       R4 - HIGH 16 BITS OF QUOTIENT
                                ;       R5 - REMAINDER
                                ;
002110                          DIVIDE: PUSH R2                                 MOV R2,-(SP)
002110  010246
002112  012702  000040                  MOV #32.,R2             ;SET UP SHIFT COUNT
002116  005005                          CLR R5                  ;START WITH ZERO REMAINDER
002120  006303                  1$:     ASL R3                  ;SHIFT LEFT INTO R5
002122  006104                          ROL R4
002124  006105                          ROL R5
002126  020005                          CMP R0,R5               ;WILL DIVISOR GO INTO REMAINDER
002130  101002                          BHI 2$                  ;ONLY SUBTRACT IF IT WILL
002132  160005                          SUB R0,R5               ;SUBTRACT DIVISOR
002134  005203                          INC R3                  ;PUT A ONE INTO QUOTIENT
002136  005302                  2$:     DEC R2                  ;COUNT THE SHIFTS
002140  001367                          BNE 1$
002142                                  POP R2                                  MOV (SP)+,R2
002142  012602
002144  000207                          RETURN
```

```
                        ;BUILD DEFAULT 28-BIT NUMBER
                        ;
                        ;INPUT:
                        ;       R4 - POINTER TO 2 WORD DEFAULT NUMBER
                        ;OUTPUT:
                        ;       TEMP - ASCIZ STRING REPRESENTING DEFAULT NUMBER

002146                  BLD28:  PUSH <R0,R1,R3,R4,R5>
002146  010046                                                          MOV R0,-(SP)
002150  010146                                                          MOV R1,-(SP)
002152  010346                                                          MOV R3,-(SP)
002154  010446                                                          MOV R4,-(SP)
002156  010546                                                          MOV R5,-(SP)
002160  011403                  MOV (R4),R3             ;GET NUMBER
002162  016404  000002          MOV 2(R4),R4
002166  012700  000012          MOV #10.,R0             ;DIVISOR IS 10.
002172  005001                  CLR R1                  ;CLEAR CHARACTER COUNT
002174  004767  177710  1$:     CALL DIVIDE
002200  062705  000060          ADD #'0,R5              ;CONVERT REMAINDER TO ASCII CHARACTER
002204                          PUSH R5                 ;STORE ON STACK
002204  010546                                                          MOV R5,-(SP)
002206  005201                  INC R1                  ;COUNT THE CHARACTER
002210  010305                  MOV R3,R5               ;REPEAT UNTIL QUOTIENT IS ZERO
002212  050405                  BIS R4,R5
002214  001367                  BNE 1$
002216  012700  004316'          MOV #TEMP,R0            ;GET POINTER TO STRING
002222          2$:     POP R5                          ;PUT CHARACTERS INTO STRING
002222  012605                                                          MOV (SP)+,R5
002224  110520                  MOVB R5,(R0)+
002226  005301                  DEC R1
002230  001374                  BNE 2$
002232  105020                  CLRB (R0)+              ;END WITH NULL
002234                          POP <R5,R4,R3,R1,R0>
002234  012605                                                          MOV (SP)+,R5
002236  012604                                                          MOV (SP)+,R4
002240  012603                                                          MOV (SP)+,R3
002242  012601                                                          MOV (SP)+,R1
002244  012600                                                          MOV (SP)+,R0
002246  000207                  RETURN
```

```
                        ;CONVERT ASCIZ STRING TO 28-BIT NUMBER
                        ;
                        ;INPUTS:
                        ;       TEMP - ASCIZ STRING UP TO 9 CHARACTERS LONK
                        ;       R4 - ADDRESS OF TWO WORD STORAGE
                        ;OUTPUTS:
                        ;       IF STRING IS VALID NUMBER
                        ;         TWO WORDS AT R4 LOADED WITH NUMBER
                        ;         R4 POINTING TO WORD AFTER STORAGE
                        ;         CARRY CLEAR
                        ;       IF STRING INVALID
                        ;         ERROR MESSAGE PRINTED
                        ;         CARRY SET
                        ;
002250                  CNV28:  PUSH <R0,R1,R2,R3>
002250  010046                                                          MOV R0,-(SP)
002252  010146                                                          MOV R1,-(SP)
002254  010246                                                          MOV R2,-(SP)
002256  010346                                                          MOV R3,-(SP)
002260  005000                  CLR R0                  ;START WITH ZEROS
002262  005001                  CLR R1
002264  012702  004316'         MOV #TEMP,R2            ;GET ADDRESS OF STRING
002270  112203          1$:     MOVB (R2)+,R3           ;GET A DIGIT FROM STRING
002272  001452                  BEQ 3$                  ;IF NULL CHARACTER, ALL DONE
002274  162703  000060          SUB #'0,R3              ;SUBTRACT CHARACTER 0
002300  100435                  BMI 2$
002302  022703  000011          CMP #9.,R3
002306  103432                  BLO 2$
002310  006300                  ASL R0                  ;MULTIPLY BY 2
002312  006101                  ROL R1
002314                          PUSH <R1,R0>            ;SAVE N X 2
002314  010146                                                          MOV R1,-(SP)
002316  010046                                                          MOV R0,-(SP)
002320  006300                  ASL R0                  ;TIMES 2 AGAIN FOR N X 4
002322  006101                  ROL R1
002324  006300                  ASL R0                  ;TIMES 2 AGAIN FOR N X 8
002326  006101                  ROL R1
002330  062600                  ADD (SP)+,R0            ;ADD N X 2 TO GIVE N X 10
002332  005501                  ADC R1
002334  062601                  ADD (SP)+,R1
002336  060300                  ADD R3,R0               ;ADD CURRENT DIGIT
002340  005501                  ADC R1
002342  032701  170000          BIT #170000,R1          ;CHECK SIZE OF NUMBER
002346  001750                  BEQ 1$                  ;MUST NOT BE MORE THAN 28 BITS
```

```
                                                PRINTF #INP28A          ;PRINT PROPER RANGE
002350  012746  004167'          MOV           #INP28A,-(SP)
002354  012746  000001          MOV           #1,-(SP)
002360  010600                  MOV           SP,R0
002362  104417                  TRAP          C$PNTF
002364  062706  000004          ADD           #4,SP
002370  000261                  SEC                                   ;SET CARRY TO ASK AGAIN
002372  000415                  BR 4$

                                2$:           PRINTF  #INP28B         ;PRINT ILLEGAL CHARACTER
002374  012746  004231'          MOV           #INP28B,-(SP)
002400  012746  000001          MOV           #1,-(SP)
002404  010600                  MOV           SP,R0
002406  104417                  TRAP          C$PNTF
002410  062706  000004          ADD           #4,SP
002414  000261                  SEC
002416  000403                  BR 4$

002420  010024                  3$:           MOV R0,(R4)+            ;MOVE NUMBER TO STORAGE AREA
002422  010124                  MOV R1,(R4)+
002424  000241                  CLC                                   ;CLEAR CARRY TO INDICATE ALL IS WELL
002426                          4$:           POP <R3,R2,R1,R0>
002426  012603                                                                MOV (SP)+,R3
002430  012602                                                                MOV (SP)+,R2
002432  012601                                                                MOV (SP)+,R1
002434  012600                                                                MOV (SP)+,R0
002436  000207                  RETURN
```

```
                                    ; UNFORMATTED QUESTIONS
                                        .NLIST BEX
 002440     116     125     115  T4BB:    .ASCIZ\NUMBER OF BAD BLOCKS\
 002465     104     117     040  T4DMN:   .ASCIZ\DO YOU WANT TO CHANGE TESTING PARAMETERS FOR THIS DRIVE\
 002555     102     101     104  T4BBI:   .ASCIZ\BAD BLOCK\
 002567     122     105     101  T4RO:    .ASCIZ\READ ONLY\
 002601     127     122     111  T4WO:    .ASCIZ\WRITE ONLY\
 002614     103     110     105  T4WCA:   .ASCIZ\CHECK ALL WRITES BY READING\
 002650     122     101     116  T4WCR:   .ASCIZ\RANDOMLY CHECK WRITES BY READING\
 002711     104     101     124  T4DP:    .ASCIZ\DATA PATTERN - 0 FOR RANDOM SELECTION\
 002757     105     116     101  T4ECC:   .ASCIZ\ENABLE ECC DATA CORRECTION\
 003012     103     117     115  T4DCA:   .ASCIZ\COMPARE ALL DATA READ\
 003040     122     101     116  T4DCR:   .ASCIZ\RANDOMLY COMPARE DATA READ\
 003073     105     116     101  T4RET:   .ASCIZ\ENABLE RETRIES\
 003112     122     101     116  T4SEK:   .ASCIZ\RANDOM ACCESS MODE\
 003135     040     040     000  T4OPT7:  .ASCIZ\  \
 003140     116     125     115  T4BE:    .ASCIZ\NUMBER OF BEGIN/END SETS\
 003171     102     105     107  T4BEG:   .ASCIZ\BEGIN BLOCK\
 003205     105     116     104  T4END:   .ASCIZ\END BLOCK\
 003217     116     125     115  T4TRC:   .ASCIZ\NUMBER OF TRACKS TO TEST\
 003250     124     122     101  T4TRAK:  .ASCIZ\TRACK\
 003256     116     125     115  T4GRC:   .ASCIZ\NUMBER OF GROUPS TO TEST\
 003307     107     122     117  T4GRP:   .ASCIZ\GROUP\
 003315     104     117     040  T4CYL:   .ASCIZ\DO YOU WISH TO LIMIT THE CYLINDERS TESTED\
 003367     123     124     101  T4CYLB:  .ASCIZ\STARTING CYLINDER\
 003411     105     116     104  T4CYLE:  .ASCIZ\ENDING CYLINDER\
 003431     116     125     115  T4DPC:   .ASCIZ\NUMBER OF WORDS IN DATA PATTERN 16\
 003474     104     101     124  T4DPD:   .ASCIZ\DATA WORD\
```

```
                                        ; FORMATTED QUESTIONS
                                        ;
003506      045     116     045 T4QHED: .ASCIZ\%N%ATHE FOLLOWING QUESTIONS REFER TO UNIT %D2%A UDA AT %O6%A DRIVE %D3%N\
003617      045     116     045 T4OPT1: .ASCIZ\%N%ADO YOU WISH TO:%N\
003645      045     101     040 T4OPT2: .ASCIZ\%A   0 - TEST ENTIRE AREA SELECTED%N\
003712      045     101     040 T4OPT3: .ASCIZ\%A   1 - SPECIFY BEGIN/END SETS TO TEST%N\
003764      045     101     040 T4OPT4: .ASCIZ\%A   2 - SPECIFY TRACKS AND CYLINDERS TO TEST%N\
004044      045     101     040 T4OPT5: .ASCIZ\%A   3 - SPECIFY GROUPS AND CYLINDERS TO TEST%N\
004124      045     101     040 T4OPT6: .ASCIZ\%A   4 - SPECIFY CYLINDERS TO TEST\
004167      045     101     114 INP28A: .ASCIZ\%ALIMITS - LO= 0, HI= 268435455%N\
004231      045     101     111 INP28B: .ASCIZ\%AINVALID CHAR, TYPE DECIMAL NUMBER 0 TO 268435455%N\
```

```
      1                                         .EVEN
      2 004316                          TEMP:   .BLKW   12
      3 004342  000000                ANYMOR: .WORD   0       ; ANY MORE QUESTIONS
      4         000001                ANY.YS = BIT0
      5         000001                         .END
```

TEST 4 QUESTIONS MACRO V04.00 18-AUG-82 11:40:09 PAGE 26-1 SEQ 0583

SYMBOL TABLE

```
ADR    = 000020 G      C$DRPT= 000024      C.UADR= 000000      D.WC  = 000010      HOE   = 100000 G
ANYMOR  004342R        C$DU  = 000053      C.UNIT= 000002      D.WCA = 000004      IBE   = 010000 G
ANY.YS= 000001         C$EDIT= 000003      C.VEC = 000004      D.WO  = 002000      IDU   = 000040 G
ASS    = 000002        C$ERDF= 000055      DDEF  = 011012      D.XFRR= 000166      IER   = 020000 G
ASSEMB= 000010         C$ERHR= 000056      DIAGMC= 000000      D.XFRW= 000164      INP28A  004167R
A1     = 000001        C$ERRO= 000060      DIVIDE  002110R     D.ZERO= 140200      INP28B  004231R
A2     = 000002        C$ERSF= 000054      DT.AVL= 100000      EF.CON= 000036 G    ISR   = 000100 G
BIT0   = 000001 G      C$ERSO= 000057      DT.UNT= 000077      EF.NEW= 000035 G    IXE   = 004000 G
BIT00  = 000001 G      C$ESCA= 000010      D.BB  = 000010      EF.PWR= 000034 G    I$AU  = 000041
BIT01  = 000002 G      C$ESEG= 000005      D.BB01= 000012      EF.RES= 000037 G    I$AUTO= 000041
BIT02  = 000004 G      C$ESUB= 000003      D.BB02= 000016      EF.STA= 000040 G    I$CLN = 000041
BIT03  = 000010 G      C$ETST= 000001      D.BB03= 000022      EN     = 040000     I$DU  = 000041
BIT04  = 000020 G      C$EXIT= 000032      D.BB04= 000026      EO     = 140000     I$INIT= 000041
BIT05  = 000040 G      C$GETB= 000026      D.BB05= 000032      EVL    = 000004 G   I$MOD = 000041
BIT06  = 000100 G      C$GETW= 000027      D.BB06= 000036      E$END  = 002100     I$MSG = 000041
BIT07  = 000200 G      C$GMAN= 000043      D.BB07= 000042      E$LOAD= 000035      I$PROT= 000041
BIT08  = 000400 G      C$GPHR= 000042      D.BB08= 000046      FS     = 100000     I$PTAB= 000041
BIT09  = 001000 G      C$GPLO= 000030      D.BB09= 000052      F$AU   = 000015     I$PWR = 000041
BIT1   = 000002 G      C$GPRI= 000040      D.BB10= 000056      F$AUTO= 000020      I$RPT = 000041
BIT10  = 002000 G      C$INIT= 000011      D.BB11= 000062      F$BGN  = 000040     I$SEG = 000041
BIT11  = 004000 G      C$INLP= 000020      D.BB12= 000066      F$CLEA= 000007      I$SETU= 000041
BIT12  = 010000 G      C$MANI= 000050      D.BB13= 000072      F$DU   = 000016     I$SRV = 000041
BIT13  = 020000 G      C$MEM = 000031      D.BB14= 000076      F$END  = 000041     I$SUB = 000041
BIT14  = 040000 G      C$MSG = 000023      D.BB15= 000102      F$HARD= 000004      I$TST = 000041
BIT15  = 100000 G      C$OPEN= 000034      D.BB16= 000106      F$HW   = 000013     J$JMP = 000167
BIT2   = 000004 G      C$PNTB= 000014      D.BCYL= 000154      F$INIT= 000006      LOE   = 040000 G
BIT3   = 000010 G      C$PNTF= 000017      D.BE  = 000040      F$JMP  = 000050     LOT   = 000010 G
BIT4   = 000020 G      C$PNTS= 000016      D.BEC = 000112      F$MOD  = 000000     MC    = 000014
BIT5   = 000040 G      C$PNTX= 000015      D.BGN1= 000114      F$MSG  = 000011     MD    = 000005
BIT6   = 000100 G      C$QIO = 000377      D.BGN2= 000124      F$PROT= 000021      MM    = 000017
BIT7   = 000200 G      C$RDBU= 000007      D.BGN3= 000134      F$PWR  = 000017     MX    = 000012
BIT8   = 000400 G      C$REFG= 000047      D.BGN4= 000144      F$RPT  = 000012     M1    = 000004
BIT9   = 001000 G      C$RESE= 000033      D.CYL = 000400      F$SEG  = 000003     M2    = 000010
BLD28    002146R       C$REVI= 000003      D.DC  = 000002      F$SOFT= 000005      O$APTS= 000000
BOE    = 000400 G      C$RFLA= 000021      D.DCA = 000001      F$SRV  = 000010     O$AU  = 000000
CNV28    002250R       C$RPT = 000025      D.DCY = 020000      F$SUB  = 000002     O$BGNR= 000000
CT.AVL= 100000         C$SEFG= 000046      D.DRV = 000000      F$SW   = 000014     O$BGNS= 000000
CT.BRL= 007000         C$SPRI= 000041      D.ECC = 010000      F$TEST= 000001      O$DU  = 000000
CT.CMD= 000004         C$SVEC= 000037      D.ECCC= 000176      G$CNTO= 000200      O$ERRT= 000000
CT.MSG= 000010         C$TPRI= 000013      D.ECYL= 000160      G$DELM= 000372      O$GNSW= 000000
CT.REQ= 000020         C.BST = 000006      D.END1= 000120      G$DISP= 000003      O$POIN= 000000
CT.RN  = 000002        C.DR0 = 000020      D.END2= 000130      G$EXCP= 000400      O$SETU= 000000
CT.UNT= 000077         C.DR1 = 000022      D.END3= 000140      G$HILI= 000002      PNT   = 001000 G
CT.VEC= 000777         C.DR2 = 000024      D.END4= 000150      G$LOLI= 000001      PRI   = 002000 G
C$AU   = 000052        C.DR3 = 000026      D.HERR= 000170      G$NO   = 000000     PRI00 = 000000 G
C$AUTO= 000061         C.DR4 = 000030      D.IW  = 040000      G$OFFS= 000400      PRI01 = 000040 G
C$BRK  = 000022        C.DR5 = 000032      D.PAT = 000006      G$OFSI= 000376      PRI02 = 000100 G
C$BSEG= 000004         C.DR6 = 000034      D.PRM = 000004      G$PRMA= 000001      PRI03 = 000140 G
C$BSUB= 000002         C.DR7 = 000036      D.RET = 001000      G$PRMD= 000002      PRI04 = 000200 G
C$CEFG= 000045         C.FLG = 000014      D.RO  = 004000      G$PRML= 000000      PRI05 = 000240 G
C$CLCK= 000062         C.JAD = 000012      D.SEEK= 000174      G$RADA= 000140      PRI06 = 000300 G
C$CLEA= 000012         C.JSR = 000010      D.SEQ = 000100      G$RADB= 000000      PRI07 = 000340 G
C$CLOS= 000035         C.REF = 000044      D.SERN= 000200      G$RADD= 000040      SVCGBL= 177777
C$CLP1= 000006         C.RING= 000016      D.SERR= 000172      G$RADL= 000120      SVCINS= 177777
C$CVEC= 000036         C.SIZE= 000046      D.SIZE= 000206      G$RADO= 000020      SVCSUB= 177777
C$DCLN= 000044         C.TO  = 000040      D.TR  = 000020      G$XFER= 000004      SVCTAG= 177777
C$DODU= 000051         C.TOH = 000042      D.UNIT= 000002      G$YES  = 000010     SVCTST= 177777
```

SYMBOL TABLE

```
S$LSYM= 010032          T$TEST= 000000          T4OPT1  003617R         T4Q07   000432R         T4Q26   001610R
TEMP    004316R         T$TSTM= 177777          T4OPT2  003645R         T4Q08   000510R         T4Q27   001670R
TINDEX= 000206          T$TSTS= 000000          T4OPT3  003712R         T4Q09   000516R         T4Q27A  001702R
T$ARGC= 000001          T4BB    002440R         T4OPT4  003764R         T4Q10   000606R         T4Q28   001710R
T$CODE= 000032          T4BBI   002555R         T4OPT5  004044R         T4Q11   000652R         T4Q29   001742R
T$ERRN= 000000          T4BE    003140R         T4OPT6  004124R         T4Q12   001050R         T4Q30   001774R
T$EXCP= 000000          T4BEG   003171R         T4OPT7  003135R         T4Q13   001104R         T4RET   003073R
T$GMAN= 000C00          T4CYL   003315R         T4PRM5  002052R         T4Q14   001136R         T4RO    002567R
T$HILI= 177777          T4CYLB  003367R         T4QHED  003506R         T4Q15   001164R         T4SEK   003112R
T$LAST= 000000          T4CYLE  003411R         T4QST   000000RG        T4Q16   001200R         T4TRAK  003250R
T$LOLI= 000000          T4DCA   003012R         T4QUES  000006R         T4Q17   001206R         T4TRC   003217R
T$LSYM= 010000          T4DCR   003040R         T4QUE2  002002R         T4Q18   001220R         T4WCA   002614R
T$NEST= 177777          T4DMN   002465R         T4QU2E  002106R         T4Q19   001226R         T4WCR   002650R
T$PTNU= 000000          T4DP    002711R         T4Q01   000112R         T4Q20   001304R         T4WO    002601R
T$SAVL= 177777          T4DPC   003431R         T4Q02   000150R         T4Q21   001336R         UAM   = 000200 G
T$SEGL= 177777          T4DPD   003474R         T4Q03   000246R         T4Q22   001376R         X$ALWA= 000000
T$SUBN= 000000          T4ECC   002757R         T4Q04   000324R         T4Q23   001464R         X$FALS= 000040
T$TAGL= 177777          T4END   003205R         T4Q05   000332R         T4Q24   001522R         X$OFFS= 000400
T$TAGN= 010000          T4GRC   003256R         T4Q06   000402R         T4Q25   001554R         X$TRUE= 000020
T$TEMP= 000010          T4GRP   003307R

. ABS.  000000      000
        004344      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  28160 WORDS  ( 110 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  70 PAGES
B:T4QUES,B:T4QUES/C=[20,0]SVC34R.MLB/P:1,ASSEM,T4QUES
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
C$ESCA   2-9#
C$ESEG   2-9#
C$ESUB   2-9#
C$ETST   2-9#
C$EXIT   2-9#
C$GETB   2-9#
C$GETW   2-9#
C$GMAN   2-9#     9-14    10-2    10-9    10-15   10-21   10-22   10-25   11-4    11-13   11-14   11-17   12-2    12-4
        13-7     15-4    15-10   15-14   16-10   16-14   16-20   17-2    17-8    17-22   18-2    19-9    19-15
C$GPHR   2-9#
C$GPLO   2-9#
C$GPRI   2-9#
C$INIT   2-9#
C$INLP   2-9#
C$MANI   2-9#
C$MEM    2-9#
C$MSG    2-9#
C$OPEN   2-9#
C$PNTB   2-9#
C$PNTF   2-9#     9-12    13-1    13-2    13-3    13-4    13-5    13-6    23-1    23-5
C$PNTS   2-9#
C$PNTX   2-9#
C$QIO    2-9#
C$RDBU   2-9#
C$REFG   2-9#
C$RESE   2-9#     2-9#
C$REVI   2-9#
C$RFLA   2-9#
C$RPT    2-9#
C$SEFG   2-9#
C$SPRI   2-9#
C$SVEC   2-9#
C$TPRI   2-9#
C.BST    5-19#
C.DR0    5-38#
C.DR1    5-39#
C.DR2    5-40#
C.DR3    5-41#
C.DR4    5-42#
C.DR5    5-43#
C.DR6    5-44#
C.DR7    5-45#
C.FLG    5-22#
C.JAD    5-21#
C.JSR    5-20#
C.REF    5-48#
C.RING   5-37#
C.SIZE   5-50#
C.TO     5-46#
C.TOH    5-47#
C.UADR   5-12#
C.UNIT   5-13#
C.VEC    5-16#
CNV28   10-3      15-11   15-15   17-23   18-3    22-15#
CT.AVL   5-15#
CT.BRL   5-18#
```

TEST 4 QUESTIONS MACRO V04.00 18-AUG-82 11:40:09 PAGE S-3 SEQ 0587
CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

TEST 4 QUESTIONS MACRO V04.00 18-AUG-82 11:40:09 PAGE S-5 SEQ 0589
CROSS REFERENCE TABLE (CREF V04.00)

TEST 4 QUESTIONS MACRO V04.00 18-AUG-82 11:40:09 PAGE S-6
CROSS REFERENCE TABLE (CREF V04.00)

SEQ 0590

PRI01 2-10#
PRI02 2-10#
PRI03 2-10#
PRI04 2-10#
PRI05 2-10#
PRI06 2-10#
PRI07 2-10#

S$LSYM 2-9# 9-14 9-14 9-14 9-14# 10-2 10-2 10-2 10-2# 10-9 10-9 10-9 10-9# 10-15
10-15 10-15 10-15# 10-21 10-21 10-21 10-21# 10-22 10-22 10-22 10-22# 10-25 10-25 10-25
10-25# 11-4 11-4 11-4 11-4# 11-13 11-13 11-13 11-13# 11-14 11-14 11-14 11-14# 11-17
11-17 11-17 11-17# 12-2 12-2 12-2 12-2# 12-4 12-4 12-4 12-4# 13-7 13-7 13-7
13-7# 15-4 15-4 15-4 15-4# 15-10 15-10 15-10 15-10# 15-14 15-14 15-14 15-14# 16-10
16-10 16-10 16-10# 16-14 16-14 16-14 16-14# 16-20 16-20 16-20 16-20# 17-2 17-2 17-2
17-2# 17-8 17-8 17-8 17-8# 17-22 17-22 17-22 17-22# 18-2 18-2 18-2 18-2# 19-9
19-9 19-9 19-9# 19-15 19-15 19-15 19-15#

SVCGBL 2-9#

SVCINS 2-9# 9-12 9-12 9-12 9-12 9-12 9-12 9-12 9-12 9-14 9-14 9-14 9-14 9-14
9-14 9-14 9-14 10-2 10-2 10-2 10-2 10-2 10-2 10-2 10-2 10-9 10-9 10-9
10-9 10-9 10-9 10-15 10-15 10-15 10-15 10-15 10-15 10-21 10-21 10-21 10-21 10-21
10-21 10-22 10-22 10-22 10-22 10-22 10-22 10-25 10-25 10-25 10-25 10-25 10-25 11-4
11-4 11-4 11-4 11-4 11-4 11-4 11-13 11-13 11-13 11-13 11-13 11-13 11-14
11-14 11-14 11-14 11-14 11-14 11-17 11-17 11-17 11-17 11-17 11-17 12-2 12-2 12-2
12-2 12-2 12-2 12-4 12-4 12-4 12-4 12-4 12-4 13-1 13-1 13-1 13-1 13-1
13-2 13-2 13-2 13-2 13-2 13-3 13-3 13-3 13-3 13-3 13-4 13-4 13-4 13-4
13-4 13-5 13-5 13-5 13-5 13-5 13-6 13-6 13-6 13-6 13-6 13-7 13-7 13-7
13-7 13-7 13-7 13-7 13-7 15-4 15-4 15-4 15-4 15-4 15-4 15-4 15-4 15-10
15-10 15-10 15-10 15-10 15-10 15-10 15-10 15-10 15-14 15-14 15-14 15-14 15-14 15-14 15-14
15-14 16-10 16-10 16-10 16-10 16-10 16-10 16-10 16-10 16-10 16-14 16-14 16-14 16-14 16-14
16-14 16-14 16-14 16-20 16-20 16-20 16-20 16-20 16-20 16-20 16-20 17-2 17-2 17-2
17-2 17-2 17-2 17-2 17-2 17-8 17-8 17-8 17-8 17-8 17-8 17-22 17-22 17-22
17-22 17-22 17-22 17-22 17-22 18-2 18-2 18-2 18-2 18-2 18-2 18-2 18-2 19-9
19-9 19-9 19-9 19-9 19-9 19-9 19-9 19-15 19-15 19-15 19-15 19-15 19-15 19-15
19-15 23-1 23-1 23-1 23-1 23-1 23-5 23-5 23-5 23-5 23-5

SVCSUB 2-9#

SVCTAG 2-9# 9-14 9-14 9-14 10-2 10-2 10-2 10-9 10-9 10-9 10-15 10-15 10-15 10-21
10-21 10-21 10-22 10-22 10-22 10-25 10-25 10-25 11-4 11-4 11-4 11-13 11-13 11-13
11-14 11-14 11-14 11-17 11-17 11-17 12-2 12-2 12-2 12-4 12-4 12-4 13-7 13-7
13-7 15-4 15-4 15-4 15-10 15-10 15-10 15-14 15-14 15-14 16-10 16-10 16-10 16-14
16-14 16-14 16-20 16-20 16-20 17-2 17-2 17-2 17-8 17-8 17-8 17-22 17-22 17-22
18-2 18-2 18-2 19-9 19-9 19-9 19-15 19-15 19-15

SVCTST 2-9#

T$ARGC 9-12 9-12 9-12 9-12 9-12 9-12# 9-12# 9-12# 9-12# 13-1 13-1 13-1# 13-2 13-2
13-2# 13-3 13-3 13-3# 13-4 13-4 13-4# 13-5 13-5 13-5# 13-6 13-6 13-6# 23-1
23-1 23-1# 23-5 23-5 23-5#

T$CODE 9-14 9-14 9-14 9-14# 9-14# 9-14# 10-2 10-2 10-2 10-2# 10-2# 10-2# 10-9 10-9
10-9 10-9# 10-9# 10-9# 10-15 10-15 10-15# 10-15# 10-15# 10-21 10-21 10-21 10-21#
10-21# 10-21# 10-22 10-22 10-22 10-22# 10-22# 10-22# 10-25 10-25 10-25 10-25# 10-25# 10-25#
11-4 11-4 11-4 11-4# 11-4# 11-4# 11-13 11-13 11-13 11-13# 11-13# 11-13# 11-14 11-14
11-14 11-14# 11-14# 11-14# 11-17 11-17 11-17 11-17# 11-17# 11-17# 12-2 12-2 12-2 12-2#
12-2# 12-2# 12-4 12-4 12-4 12-4# 12-4# 12-4# 13-7 13-7 13-7 13-7# 13-7# 13-7#
15-4 15-4 15-4 15-4# 15-4# 15-4# 15-10 15-10 15-10 15-10# 15-10# 15-10# 15-14 15-14
15-14 15-14# 15-14# 15-14# 16-10 16-10 16-10 16-10# 16-10# 16-10# 16-14 16-14 16-14 16-14#
16-14# 16-14# 16-20 16-20 16-20 16-20# 16-20# 16-20# 17-2 17-2 17-2 17-2# 17-2# 17-2#
17-8 17-8 17-8 17-8# 17-8# 17-8# 17-22 17-22 17-22 17-22# 17-22# 17-22# 18-2 18-2
18-2 18-2# 18-2# 18-2# 19-9 19-9 19-9 19-9# 19-9# 19-9# 19-15 19-15 19-15 19-15#
19-15# 19-15#

TEST 4 QUESTIONS MACRO V04.00 18-AUG-82 11:40:09 PAGE S-7 SEQ 0591
CROSS REFERENCE TABLE (CREF V04.00)

T$ERRN	2-9#													
T$EXCP	9-14	9-14#	10-2	10-2#	11-4	11-4#	13-7	13-7#	15-4	15-4#	15-10	15-10#	15-14	15-14#
	16-10	16-10#	16-14	16-14#	16-20	16-20#	17-2	17-2#	17-22	17-22#	18-2	18-2#	19-9	19-9#
	19-15	19-15#												
T$GMAN	2-9#	9-14#	9-14#	10-2	10-2#	10-2#	11-4#	11-4#	13-7#	13-7#	15-4#	15-4#	15-10	15-10#
	15-10#	15-14	15-14#	15-14#	16-10#	16-10#	16-14#	16-14#	16-20#	16-20#	17-2#	17-2#	17-22	17-22#
	17-22#	18-2	18-2#	18-2#	19-9#	19-9#	19-15#	19-15#						
T$HILI	9-14	9-14#	10-2	10-2#	11-4	11-4#	13-7	13-7#	15-4	15-4#	15-10	15-10#	15-14	15-14#
	16-10	16-10#	16-14	16-14#	16-20	16-20#	17-2	17-2#	17-22	17-22#	18-2	18-2#	19-9	19-9#
	19-15	19-15#												
T$LAST	2-9#													
T$LOLI	9-14	9-14#	10-2	10-2#	11-4	11-4#	13-7	13-7#	15-4	15-4#	15-10	15-10#	15-14	15-14#
	16-10	16-10#	16-14	16-14#	16-20	16-20#	17-2	17-2#	17-22	17-22#	18-2	18-2#	19-9	19-9#
	19-15	19-15#												
T$LSYM	2-9	2-9#												
T$NEST	2-9#													
T$PTNU	2-9#													
T$SAVL	2-9#													
T$SEGL	2-9#													
T$SUBN	2-9#													
T$TAGL	2-9#													
T$TAGN	2-9#													
T$TEMP	9-14	9-14	9-14	9-14#	9-14#	9-14#	10-2	10-2	10-2	10-2#	10-2#	10-2#	10-9	10-9
	10-9	10-9#	10-9#	10-9#	10-15	10-15	10-15	10-15#	10-15#	10-15#	10-21	10-21	10-21	10-21#
	10-21#	10-21#	10-22	10-22	10-22	10-22#	10-22#	10-22#	10-25	10-25	10-25	10-25#	10-25#	10-25#
	11-4	11-4	11-4	11-4#	11-4#	11-4#	11-13	11-13	11-13	11-13#	11-13#	11-13#	11-14	11-14
	11-14	11-14#	11-14#	11-14#	11-17	11-17	11-17	11-17#	11-17#	11-17#	12-2	12-2	12-2	12-2#
	12-2#	12-2#	12-4	12-4	12-4	12-4#	12-4#	12-4#	13-7	13-7	13-7	13-7#	13-7#	13-7#
	15-4	15-4	15-4	15-4#	15-4#	15-4#	15-10	15-10	15-10	15-10#	15-10#	15-10#	15-14	15-14
	15-14	15-14#	15-14#	15-14#	16-10	16-10	16-10	16-10#	16-10#	16-10#	16-14	16-14	16-14	16-14#
	16-14#	16-14#	16-20	16-20	16-20	16-20#	16-20#	16-20#	17-2	17-2	17-2	17-2#	17-2#	17-2#
	17-8	17-8	17-8	17-8#	17-8#	17-8#	17-22	17-22	17-22	17-22#	17-22#	17-22#	18-2	18-2
	18-2	18-2#	18-2#	18-2#	19-9	19-9	19-9	19-9#	19-9#	19-9#	19-15	19-15	19-15	19-15#
	19-15#	19-15#												
T$TEST	2-9#													
T$TSTM	2-9#	9-12	9-14	10-2	10-9	10-15	10-21	10-22	10-25	11-4	11-13	11-14	11-17	12-2
	12-4	13-1	13-2	13-3	13-4	13-5	13-6	13-7	15-4	15-10	15-14	16-10	16-14	16-20
	17-2	17-8	17-22	18-2	19-9	19-15	23-1	23-5						
T$TSTS	2-9#													
T4BB	9-14	24-3#												
T4BBI	10-2	24-5#												
T4BE	15-4	24-7#												
T4BEG	15-10	24-18#												
T4CYL	17-8	24-24#												
T4CYLB	17-22	24-25#												
T4CYLE	18-2	24-26#												
T4DCA	11-14	24-12#												
T4DCR	11-17	24-13#												
T4DMN	10-9	24-4#												
T4DP	11-4	24-10#												
T4DPC	19-9	24-27#												
T4DPD	19-15	24-28#												
T4ECC	11-13	24-11#												
T4END	15-14	24-19#												
T4GRC	16-20	24-22#												
T4GRP	17-2	24-23#												

CROSS REFERENCE TABLE (CREF V04.00)

T4OPT1	13-1	25-4#											
T4OPT2	13-2	25-5#											
T4OPT3	13-3	25-6#											
T4OPT4	13-4	25-7#											
T4OPT5	13-5	25-8#											
T4OPT6	13-6	25-9#											
T4OPT7	13-7	24-16#											
T4PRM5	19-14#	19-18											
T4Q01	10-1#	10-4	10-6										
T4Q02	9-16	10-7#											
T4Q03	10-17	10-20#											
T4Q04	10-24	11-1#											
T4Q05	10-19	10-26	11-2#										
T4Q06	11-7#												
T4Q07	11-8	11-10	11-13#										
T4Q08	11-16	12-1#											
T4Q09	11-12	11-18	12-2#										
T4Q10	12-10	12-13#											
T4Q11	12-12	12-14	12-17	12-20	13-1#								
T4Q12	13-9	14-1#											
T4Q13	14-2	14-8#											
T4Q14	14-11	14-15#											
T4Q15	14-18	14-20#											
T4Q16	14-7	14-21	14-24#										
T4Q17	14-25#												
T4Q18	14-28#	14-30											
T4Q19	13-11	14-4	14-14	14-23	15-1#								
T4Q20	15-9#	15-12	15-18										
T4Q21	15-13#	15-16											
T4Q22	15-2	16-1#											
T4Q23	16-13#	16-17											
T4Q24	16-8	16-19#											
T4Q25	17-1#	17-5											
T4Q26	16-18	17-6#											
T4Q27	16-4	17-10	17-16#										
T4Q27A	17-17	17-19#											
T4Q28	17-21#	17-24											
T4Q29	18-1#	18-4											
T4Q30	10-12	15-19	16-2	17-15	18-5#								
T4QHED	9-12	25-3#											
T4QST	2-6	8-2#											
T4QU2E	19-7	19-13	19-19#										
T4QUE2	8-5	19-5#											
T4QUES	8-3	9-11#											
T4RET	12-2	24-14#											
T4RO	10-15	24-6#											
T4SEK	12-4	24-15#											
T4TRAK	16-14	24-21#											
T4TRC	16-10	24-20#											
T4WCA	10-22	24-8#											
T4WCR	10-25	24-9#											
T4WO	10-21	24-7#											
TEMP	9-13*	9-14	9-15	9-20	10-2	10-14*	10-15	10-16	10-18*	10-21	10-22	10-23	10-25	11-1*
	11-2	11-3*	11-4	11-5	11-6*	11-7	11-9	11-11	11-13	11-14	11-15	11-17	12-1*	12-2
	12-3*	12-4	12-5*	12-6	12-8*	12-11*	12-15*	12-18*	12-21*	13-7	13-8*	14-1*	14-10	14-17*
	15-3*	15-4	15-5	15-6	15-10	15-14	16-9*	16-10	16-11	16-13*	16-14	16-15	16-19*	16-20

TEST 4 QUESTIONS MACRO V04.00 18-AUG-82 11:40:09 PAGE S-9
CROSS REFERENCE TABLE (CREF V04.00)

SEQ 0593

	16-21	17-1*	17-2	17-3	17-6*	17-7*	17-8	17-9	17-22	18-2	19-8*	19-9	19-10	19-12
	19-14*	19-15	19-16	21-20	22-18	26-2#								
TINDEX	5-10#	5-12	5-12	5-12#	5-13	5-13	5-13#	5-16	5-16	5-16#	5-19	5-19	5-19#	5-20
	5-20	5-20#	5-21	5-21	5-21#	5-22	5-22	5-22#	5-37	5-37	5-37#	5-38	5-38	5-38#
	5-39	5-39	5-39#	5-40	5-40	5-40#	5-41	5-41	5-41#	5-42	5-42	5-42#	5-43	5-43
	5-43#	5-44	5-44	5-44#	5-45	5-45	5-45#	5-46	5-46	5-46#	5-47	5-47	5-47#	5-48
	5-48	5-48#	5-50	6-7#	6-9	6-9	6-9#	6-10	6-10	6-10#	6-13	6-13	6-13#	6-40
	6-40	6-40#	6-41	6-41	6-41#	6-42	6-42	6-42#	6-43	6-43	6-43#	6-44	6-44	6-44#
	6-45	6-45	6-45#	6-46	6-46	6-46#	6-47	6-47	6-47#	6-48	6-48	6-48#	6-49	6-49
	6-49#	6-50	6-50	6-50#	6-51	6-51	6-51#	6-52	6-52	6-52#	6-53	6-53	6-53#	6-54
	6-54	6-54#	6-55	6-55	6-55#	6-56	6-56	6-56#	6-57	6-57	6-57#	7-1	7-1	7-1#
	7-2	7-2	7-2#	7-3	7-3	7-3#	7-4	7-4	7-4#	7-5	7-5	7-5#	7-6	7-6
	7-6#	7-7	7-7	7-7#	7-8	7-8	7-8#	7-9	7-9	7-9#	7-10	7-10	7-10#	7-11
	7-11	7-11#	7-12	7-12	7-12#	7-13	7-13	7-13#	7-14	7-14	7-14#	7-15	7-15	7-15#
	7-16	7-16	7-16#	7-17	7-17	7-17#	7-18	7-18	7-18#	7-27				
UAM	2-10#													
XSALWA	2-9#													
XSFALS	2-9#													
XSOFFS	2-9#													
XSTRUE	2-9#													

TEST 4 QUESTIONS MACRO V04.00 18-AUG-82 11:40:09 PAGE M-1 SEQ 0594
CROSS REFERENCE TABLE (CREF V04.00)

	15-4	15-4#	15-4#	15-4#	15-4#	15-10	15-10	15-10	15-10	15-10	15-10	15-10	15-10	15-10#
	15-10#	15-10#	15-10#	15-14	15-14	15-14	15-14	15-14	15-14	15-14	15-14	15-14#	15-14#	15-14#
	15-14#	16-10	16-10	16-10	16-10	16-10	16-10	16-10	16-10	16-10#	16-10#	16-10#	16-10#	16-14
	16-14	16-14	16-14	16-14	16-14	16-14	16-14	16-14#	16-14#	16-14#	16-14#	16-20	16-20	16-20
	16-20	16-20	16-20	16-20	16-20	16-20#	16-20#	16-20#	16-20#	17-2	17-2	17-2	17-2	17-2
	17-2	17-2	17-2	17-2#	17-2#	17-2#	17-2#	17-8	17-8	17-8	17-8	17-8	17-8	17-8#
	17-8#	17-8#	17-8#	17-22	17-22	17-22	17-22	17-22	17-22	17-22	17-22	17-22#	17-22#	17-22#
	17-22#	18-2	18-2	18-2	18-2	18-2	18-2	18-2	18-2	18-2#	18-2#	18-2#	18-2#	19-9
	19-9	19-9	19-9	19-9	19-9	19-9	19-9	19-9#	19-9#	19-9#	19-9#	19-15	19-15	19-15
	19-15	19-15	19-15	19-15	19-15	19-15#	19-15#	19-15#	19-15#	23-1	23-1	23-1	23-1	23-1
	23-1#	23-1#	23-1#	23-1#	23-5	23-5	23-5	23-5	23-5	23-5#	23-5#	23-5#	23-5#	
MSGNLS	9-14	9-14#	10-2	10-2#	10-9	10-9#	10-15	10-15#	10-21	10-21#	10-22	10-22#	10-25	10-25#
	11-4	11-4#	11-13	11-13#	11-14	11-14#	11-17	11-17#	12-2	12-2#	12-4	12-4#	13-7	13-7#
	15-4	15-4#	15-10	15-10#	15-14	15-14#	16-10	16-10#	16-14	16-14#	16-20	16-20#	17-2	17-2#
	17-8	17-8#	17-22	17-22#	18-2	18-2#	19-9	19-9#	19-15	19-15#				
MSINCR	9-12#	9-14	9-14#	9-14#	10-2	10-2#	10-2#	10-9	10-9#	10-9#	10-15	10-15#	10-15#	10-21
	10-21#	10-21#	10-22	10-22#	10-22#	10-25	10-25#	10-25#	11-4	11-4#	11-4#	11-13	11-13#	11-13#
	11-14	11-14#	11-14#	11-17	11-17#	11-17#	12-2	12-2#	12-2#	12-4	12-4#	12-4#	13-1#	13-2#
	13-3#	13-4#	13-5#	13-6#	13-7	13-7#	13-7#	15-4	15-4#	15-4#	15-10	15-10#	15-10#	15-14
	15-14#	15-14#	16-10	16-10#	16-10#	16-14	16-14#	16-14#	16-20	16-20#	16-20#	17-2	17-2#	17-2#
	17-8	17-8#	17-8#	17-22	17-22#	17-22#	18-2	18-2#	18-2#	19-9	19-9#	19-9#	19-15	19-15#
	19-15#	23-1#	23-5#											
MSMCHI	2-9	2-9#												
MSMCLO	2-9	2-9#												
MSPRIN	9-12	9-12#	13-1	13-1#	13-2	13-2#	13-3	13-3#	13-4	13-4#	13-5	13-5#	13-6	13-6#
	23-1	23-1#	23-5	23-5#										
MSPUT	9-12	9-12	9-12	9-12	9-12	9-12#	13-1	13-1	13-1#	13-2	13-2	13-2#	13-3	13-3
	13-3#	13-4	13-4	13-4#	13-5	13-5	13-5#	13-6	13-6	13-6#	23-1	23-1	23-1#	23-5
	23-5	23-5#												
MSPUT1	9-12	9-12	9-12	9-12	9-12	9-12#	9-12#	9-12#	9-12#	9-12#	13-1	13-1	13-1#	13-1#
	13-2	13-2	13-2#	13-2#	13-3	13-3	13-3#	13-3#	13-4	13-4	13-4#	13-4#	13-5	13-5
	13-5#	13-5#	13-6	13-6	13-6#	13-6#	23-1	23-1	23-1#	23-1#	23-5	23-5	23-5#	23-5#
MSRADI	9-14	9-14#	10-2	10-2#	10-9	10-9#	10-15	10-15#	10-21	10-21#	10-22	10-22#	10-25	10-25#
	11-4	11-4#	11-13	11-13#	11-14	11-14#	11-17	11-17#	12-2	12-2#	12-4	12-4#	13-7	13-7#
	15-4	15-4#	15-10	15-10#	15-14	15-14#	16-10	16-10#	16-14	16-14#	16-20	16-20#	17-2	17-2#
	17-8	17-8#	17-22	17-22#	18-2	18-2#	19-9	19-9#	19-15	19-15#				
MSSVC	9-12	9-12#	9-14	9-14#	10-2	10-2#	10-9	10-9#	10-15	10-15#	10-21	10-21#	10-22	10-22#
	10-25	10-25#	11-4	11-4#	11-13	11-13#	11-14	11-14#	11-17	11-17#	12-2	12-2#	12-4	12-4#
	13-1	13-1#	13-2	13-2#	13-3	13-3#	13-4	13-4#	13-5	13-5#	13-6	13-6#	13-7	13-7#
	15-4	15-4#	15-10	15-10#	15-14	15-14#	16-10	16-10#	16-14	16-14#	16-20	16-20#	17-2	17-2#
	17-8	17-8#	17-22	17-22#	18-2	18-2#	19-9	19-9#	19-15	19-15#	23-1	23-1#	23-5	23-5#
MSTLAB	9-12#	9-14#	10-2#	10-9#	10-15#	10-21#	10-22#	10-25#	11-4#	11-13#	11-14#	11-17#	12-2#	12-4#
	13-1#	13-2#	13-3#	13-4#	13-5#	13-6#	13-7#	15-4#	15-10#	15-14#	16-10#	16-14#	16-20#	17-2#
	17-8#	17-22#	18-2#	19-9#	19-15#	23-1#	23-5#							
MSTSTL	9-12	9-12#	9-14	9-14#	10-2	10-2#	10-9	10-9#	10-15	10-15#	10-21	10-21#	10-22	10-22#
	10-25	10-25#	11-4	11-4#	11-13	11-13#	11-14	11-14#	11-17	11-17#	12-2	12-2#	12-4	12-4#
	13-1	13-1#	13-2	13-2#	13-3	13-3#	13-4	13-4#	13-5	13-5#	13-6	13-6#	13-7	13-7#
	15-4	15-4#	15-10	15-10#	15-14	15-14#	16-10	16-10#	16-14	16-14#	16-20	16-20#	17-2	17-2#
	17-8	17-8#	17-22	17-22#	18-2	18-2#	19-9	19-9#	19-15	19-15#	23-1	23-1#	23-5	23-5#
MSWORD	9-14	9-14	9-14#	9-14#	10-2	10-2	10-2#	10-2#	10-9	10-9	10-9#	10-9#	10-15	10-15
	10-15#	10-15#	10-21	10-21	10-21#	10-21#	10-22	10-22	10-22#	10-22#	10-25	10-25	10-25#	10-25#
	11-4	11-4	11-4#	11-4#	11-13	11-13	11-13#	11-13#	11-14	11-14	11-14#	11-14#	11-17	11-17
	11-17#	11-17#	12-2	12-2	12-2#	12-2#	12-4	12-4	12-4#	12-4#	13-7	13-7	13-7#	13-7#
	15-4	15-4	15-4#	15-4#	15-10	15-10	15-10#	15-10#	15-14	15-14	15-14#	15-14#	16-10	16-10	
	16-10#	16-10#	16-14	16-14	16-14#	16-14#	16-20	16-20	16-20#	16-20#	17-2	17-2	17-2#	17-2#
	17-8	17-8	17-8#	17-8#	17-22	17-22	17-22#	17-22#	18-2	18-2	18-2#	18-2#	19-9	19-9

TABLE OF CONTENTS

TABLE OF CONTENTS

UDAT4 DISK EXERCISER DMACR X04.01 23-AUG-82 12:06:46 SEQ 0598

TABLE OF CONTENTS

```
                                .TITLE  UDAT4   DISK EXERCISER
                        ;
                        ;
                        ; COPYRIGHT (C)  1981,1982
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;THIS PROGRAM SHALL BE ASSEMBLED WITH THE PROGRAM DMACRO
                        ;USING A COMMAND LINE SIMILAR TO:
                        ;
                        ;       UDAT4.BIN,UDAT4/C=$DMACRO,UDAT4
                        ;
                        ;       ONE, AND ONLY ONE OF THE FOLLOWING ASSEMBLY OPTIONS <<MUST>> BE
                        ;       SELECTED
                        ;
000000                  UDA50   =       0       ; SET TO 1 IF ASSEMBLY IS FOR UDA50
000001                  UDA52   =       1       ; SET TO 1 IF ASSEMBLY IS FOR UDA52
000000                  QDA     =       0       ; SET TO 1 IF ASSEMBLY IS FOR QDA
                        ;
                        ;
                        ;       OTHER OPTIONS
                        ;
000000                  DEBUG   =       0       ; UNTIL TESTED, MUST BE 0 FOR UDA52
000000                  INCODT  =       0       ; SET TO 1 TO INCLUDE UDA52 ODT
                        ;       .ENABL  ABS
                                .MCALL  DMCODE,DMEND,DMOVLY
                                .MCALL  JMP,BR,BEQ,CALL,BPL,BCC,BNE,BMI,RETURN
                                .MCALL  DMODT

000000                          DMCODE  UDADM4,0,1364,3,0,1000
```

MODIFICATION CONTROL

```
                                .SBTTL  MODIFICATION CONTROL
000000                                  .REPT   0


                                MODIFICATION CONTROL -- ALL CHANGES AFTER OCTOBER 29, 1981 WILL BE
                                ENTERED IN THIS SECTION TO ALLOW THE HISTORY OF TEST 4 TO BE DOCUMENTED
                                JULY , 1982
                                        THE ''GET STATUS AFTER ERROR'' CODE (IN ROUTINE CORECT)
                                        WAS CHANGED TO OVERCOME THE RA81 PROBLEM WHERE RECEIVER
                                        READY WAS NOT IMMEDIATELY ASSERTED AFTER AN ERROR (THE
                                        DRIVE WAS ''PICKING UP THE PIECES'' AFTER AN ERROR, AND
                                        ALL F'S USED TO BE DISPLAYED BECAUSE I WOULDN'T SEE
                                        RECEIVER READY WHEN I WANTED TO).  THIS REQUIRED MOVING
                                        THE ''WAIT FOR CLOCKS'' CODE, WHICH USED TO BE IN THE
                                        ''GET STATUS AFTER ERROR'' CODE (IN CORECT) TO THE MODULE
                                        GOINIT, AND REWRITING A SMALL AMOUNT OF CODE IN CORECT.

                                        ALL CODE ADDED FOR CONDITIONAL ASSEMBLY FOR THE
                                        UDA50/UDA52.
                                .ENDR
```

```
                                .SBTTL  UDA DM PROGRAM PARAMETERS

                                        .LIST MEB
                                ;
                                ;       EQUATES
                                ;
                                ;       HIGHEST USABLE LOCATION OF UDA MEMORY + 1
                                ;
037777          HIMEM   =       37777           ; HIGH MEMORY+1
001362          OVSTRT  =       OVE.MN-2        ; OVERLAY ADDRESS LOCATION
                                ;
                                ;
                                ;       OFFSETS FOR FORMAT TRACK TABLE
                                ;
000000          FT.BUF  =       0.              ; BUFFER POINTER OFFSET
000001          FT.HI   =       1.              ; HI ORDER HEADER OFFSET
000002          FT.LOW  =       2.              ; LOW ORDER HEADER OFFSET
                                ;
                                ;
                                ;       OFFSETS FOR FORMAT TRACK BUFFER
                                ;
000000          FB.DAT  =         0.            ; FIRST DATA WORD OFFSET
000400          FB.EDC  =       256.            ; EDC WORD OFFSET
                                ;
                                ;
                                ;       OFFSETS FOR READ/WRITE I/O CHAIN TABLES
                                ;
000000          RW.CPT  =       0.              ; NEXT BUFFER POINTER OFFSET
000001          RW.STAT =       RW.CPT+1.       ; STATUS
000002          RW.BUF  =       RW.STAT+1       ; POINTER TO DATA BUFFER
000003          RW.LOW  =       RW.BUF+1        ; HI ORDER EXPECTED HEADER
000004          RW.HI   =       RW.LOW+1        ; LOW ORDER EXPECTED HEADER
000005          RW.CMD  =       RW.HI+1         ; SDI COMMAND AND HEAD ADDRESS
000006          RW.SDI  =       RW.CMD+1        ; DUMMY SDI CONTROL BLOCK POINTER
000007          RW.ANG  =       RW.SDI+1        ; THETA FROM INDEX
                                ;
                                ;       CONSTANTS FOR READ AND WRITE XFC'S
                                ;
040000          BUFFLG  =       40000           ; BUFFER FULL OR EMPTY FLAG
010000          ECCFLG  =       10000           ; ECC ERROR IN BUFFER BIT
100000          EOC     =       100000          ; END OF CHAIN
100000          FSTOP   =       100000          ; LAST ENTRY IN CHAIN FOR FORMAT
000000          RCONT   =       0               ; READ CONTINUE
013400          RREAL   =       13400           ; READ REAL TIME COMMAND
100000          RSTOP   =       EOC             ; LAST ENTRY IN CHAIN FOR READ
040000          WCONT   =       40000           ; WRITE CONTINUE
122400          WREAL   =       122400          ; WRITE REAL TIME ECOMMAND
140000          WSTOP   =       EOC+BUFFLG      ; LAST ENTRY IN CHAIN FOR WRI-
000400          SEC512  =       256.            ; 256 WORDS IN A SECTOR IN 51- BYTE MODE
000440          SEC576  =       288.            ; 288 WORDS IN A SECTOR IN 576 BYTE MODE
                                ;
                                ;
                                ;       HEADER CODES
                                ;
000000          HD.LBN  =       000000          ; GOOD LBN
060000          HD.RBN  =       060000          ; GOOD RBN, PERHAPS UNUSED
030000          HD.REV  =       030000          ; REVECTORED LBN
```

UDA DM PROGRAM PARAMETERS

```
     81          110000                  HD.BAD  =       110000          ; BAD BLOCK
     82          050000                  HD.PRV  =       050000          ; PRIMARY REVECTORED BLOCK
     83          120000                  HD.XBN  =       120000          ; XBN BLOCK
     84          140000                  HD.DBN  =       140000          ; DBN BLOCK
     85
     86                                  ;       OFFSETS FOR DATA BUFFERS
     87
     88          000000                  BF.DAT  =         0.            ; DATA
     89          000400                  BF.EDC  =       256.            ; ERROR DETECTION CODE
     90          000401                  BF.ECC  =       257.            ; LAST 12 ECC RESIDUES
     91
     92                                  ;       BUFFER AND READ/WRITE CHAIN LINK SIZES
     93
     94          000401                  WBUFLN  =       257.            ; WRITE BUFFER SIZE
     95          000415                  RBUFLN  =       WBUFLN+12.      ; READ BUFFER SIZE
    104          000010                  LINKLN  =       8.
```

```
                                ;       XFC DEFINITION EQUATES
                                ;
000000                          BREAK   =       0.              ; BREAKPOINT XFC CODE
000001                          FORMAT  =       1.              ; FORMAT TRACK XFC CODE
000002                          XREAD   =       2.              ; READ N SECTORS XFC CODE
000003                          XWRITE  =       3.              ; WRITE N SECTORS XFC CODE
000004                          SEND    =       4.              ; SEND SDI COMMAND XFC CODE
000005                          RCV     =       5.              ; RECEIVE SDI MESSAGE XFC CODE
000006                          COMPARE =       6.              ; COMPARE DATA PATTERN TO BUFFER
000007                          STATUS  =       7.              ; RETURN DRIVE STATUS XFC CODE
000010                          ECHO    =       8.              ; ECHO DATA TO DRIVE XFC CODE
000011                          DINIT   =       9.              ; DRIVE INITIALIZE XFC CODE
000012                          WAITSI  =       10.             ; WAIT FOR SECTOR OR INDEX PULSE
000013                          UREAD   =       11.             ; READ UNIBUS MEMORY XFC CODE
000014                          UWRITE  =       12.             ; WRITE UNIBUS MEMORY XFC CODE
000015                          ECC     =       13.             ; DO ECC ON BUFFER XFC CODE
000016                          MRD     =       14.             ; SEND BUFFER TO MAINTENANCE READ COMMAND
000017                          MWR     =       15.             ; GET BUFFER FROM MAINTENANCE WRITE COMMAND
000020                          CVT     =       16.             ; CONVERT TO PHYSICAL ADDRESS XFC CODE
000021                          EXIT    =       17.             ; TERMINATE DM PROGRAM XFC CODE
                                ;
                                ;       MEDIA TYPES
                                ;
                                ;       THIS WORD IS FOUND AS THE FIRST WORD OF XBN 0 (OR ANY OF ITS COPIES)
                                ;
126736                          MOD512  =       126736          ; 512 BYTE MODE
074161                          MOD576  =       074161          ; 576 BYTE MODE
                                ;
                                ;       GET STATUS OFFSETS
                                ;
000000                          ST.UNT  =       0.              ; UNIT NUMBER
000000                          ST.MSK  =       0.              ; SUBUNIT MASK
000001                          ST.REQ  =       1.              ; REQUEST BYTE
000001                          ST.MOD  =       1.              ; MODE BYTE
000002                          ST.ERR  =       2.              ; ERROR BYTE
000002                          ST.CON  =       2.              ; CONTROLLER BYTE
000002                          ST.C    =       2.              ; C BITS
000003                          ST.RTY  =       3.              ; RETRY COUNT/FAILURE CODE
100000                          VALID   =       100000          ; USED IN ERROR POSITION TO MARK VALID STATUS
                                                                ;  IN INPUT BUFFER
                                ;
                                ;       STATUS BIT DEFINITIONS
                                ;
000200                          ST.OA   =       200             ; ONLINE TO ANOTHER (SET IF DRIVE UNAVAILABLE)
000100                          ST.RR   =       100             ; READJUSTMENT BIT (SET IF RECALIBRATION REQUIRED)
000040                          ST.DR   =       40              ; DIAGNOSTIC REQUEST (SET IF DIAGNOSTIC REQUESTED)
000020                          ST.SR   =       20              ; SPINDLE READY (SET IF SPINDLE READY)
000010                          ST.EL   =       10              ; LOGGABLE INFO IN EXTENDED STATUS
000002                          ST.PS   =       2               ; PORT SWITCH (SET IF PORT SWITCH IN)
000001                          ST.RU   =       1               ; RUN/STOP SWITCH (SET IF RUN/STOP SWITCH IN)
000200                          ST.DE   =       200             ; DRIVE ERROR
000100                          ST.RE   =       100             ; RETRIABLE ERROR (SET IF RETRIABLE ERROR OCCURRED)
000040                          ST.PE   =       40              ; PROTOCOL ERROR (SET IF PROTOCOL ERROR OCCURRED)
000020                          ST.DF   =       20              ; INITIALIZATION FAILURE (SET IF INIT FAILED)
000010                          ST.WE   =       10              ; WRITE ENABLE (SET IF WRT ATTEMPTED ON PROT DISK)
002000                          ST.FO   =       2000            ; FORMATTING (SET IF FORMATTING ENABLED)
001000                          ST.DB   =       1000            ; DIAGNOSTIC CYLS (SET IF DIAG CYL ACCESS ENABLED)
```

```
     58          000400                    ST.S7   =       400          ; SECTOR SIZE (SET FOR 576 BYTE SECTORS)
```

```
                                ;       GET COMMON CHARACTERISTICS OFFSETS
                                ;
000000                          SHRTTO  =       0.              ; SHORT TIMEOUT <3:0>
000000                          SDIVER  =       0.              ; SDI VERSION <7:4>
000000                          XFERRT  =       0.              ; TRANSFER RATE <15:0>
000001                          LONGTO  =       1.              ; LONG TIMEOUT <3:0>
000001                          RETS    =       1.              ; RETRIES <7:4>
000001                          RCTCPS  =       1.              ; F/RCT COPIES <11:8>
000001                          SS      =       1.              ; SECTOR SIZE <15:15>
000002                          ERLEV   =       2.              ; ERROR RETRY LEVELS <7:0>
000002                          ECCRSH  =       2.              ; ECC THRESHOLD <15:8>
000003                          MICREV  =       3.              ; MICROCODE REVISION NUMBER <7:0>
000003                          HRDREV  =       3.              ; HARDWARE REVISION NUMBER <15:8>
000004                          DRVID   =       4.              ; UNIQUE DRIVE ID <47:0>
000007                          DRTYPE  =       7.              ; DRIVE TYPE IDENTIFIER <7:0>
000007                          REVS    =       7.              ; REVS/SECOND <15:8>
                                ;
                                ;       GET SUBUNIT CHARACTERISTICS OFFSETS
                                ;
                                ;THESE OFFSETS ARE CURRENTLY GIVEN AS FOLLOWING THE COMMON CHARACTERISTICS
                                ;
000000                          LBNCYL  =       0.              ; NUMBER OF CYLINDERS IN LBN AREA <31:0>
000001                          HICYL   =       1.              ; HI ORDER CYLINDER BITS <15:12>
000002                          GRPCYL  =       2.              ; GROUPS PER CYLINDER <7:0>
000002                          HILBN   =       2.              ; HI STARTING LBN <11:8>
000002                          HIXBN   =       2.              ; HI STARTING XBN <15:12>
000003                          TRKGRP  =       3.              ; TRACKS PER GROUP <7:0>
000003                          HIRBN   =       3.              ; HI STARTING RBN <11:8>
000003                          HIDBN   =       3.              ; HI STARTING DBN <15:12>
000004                          RBNTRK  =       4.              ; RBNS PER TRACK <6:0>
000004                          RM      =       4.              ; REMOVABLE MEDIA <7:7> 1=REMOVEABLE
000005                          DATPRE  =       5.              ; DATA PREAMBLE SIZE IN WORDS <7:0>
000005                          HDRPRE  =       5.              ; HEADER PREAMBLE SIZE IN WORDS <15:8>
000006                          MEDTYP  =       6.              ; MEDIA TYPE <31:0>
000010                          FCTSIZ  =       8.              ; FCT COPY SIZE <15:0>
000011                          LBNTRK  =       9.              ; LBNS PER TRACK <7:0>
000011                          GRPOFF  =       9.              ; GROUP OFFSET (SECTORS) <15:8>
000012                          LBNHST  =       10.             ; LBNS IN HOST AREA <31:0>
000014                          RCTCSZ  =       12.             ; RCT COPY SIZE <15:0>
000021                          XBNCYL  =       17.             ; CYLS IN XBN AREA <15:0>
000022                          DBNCYL  =       18.             ; CYLS IN DBN AREA <15:8>
                                ;
                                ;
                                ;       BIT MASK DEFINITIONS
                                ;
                                ;
177400                          HIBYTE  =       177400          ; HIGH BYTE MASK
000377                          LOBYTE  =       000377          ; LOW BYTE MASK
007777                          HBHINB  =       7777            ; HI BYTE, HI NIBBLE MASK
170377                          HBLONB  =       170377          ; HI BYTE, LO NIBBLE MASK
177417                          LBHINB  =       177417          ; LO BYTE, HI NIBBLE MASK
177760                          LBLONB  =       177760          ; LO BYTE, LO NIBBLE MASK
140000                          UNADDR  =       140000          ; UNUSED ADDRESSING BITS
                                ;
```

```
                                ;MAINTANENCE READ/WRITE REQUEST NUMBERS
                                ;
060000                  DUPPKT  =       060000          ; DUP SPECIAL PACKET NUMBER
060000                  T1MSIZ  =       0.+DUPPKT       ; GET FREE MEMORY PARAMETERS
060001                  T2DLL   =       1.+DUPPKT       ; DOWNLINE LOAD DRIVE DIAGNOSTIC
060002                  T2CMD   =       2.+DUPPKT       ; MANUAL INTERVENTION TEST 2 PROTOCOL
060003                  T4MPRM  =       3.+DUPPKT       ; GET MASTER PARAMETERS FROM SW QUESTIONS
060004                  T4UPRM  =       4.+DUPPKT       ; GET UNIT PARAMETERS FROM HW QUESTIONS
060005                  T4BB1   =       5.+DUPPKT       ; GET BAD BLOCKS (1 THRU 14)
060006                  T4BB2   =       6.+DUPPKT       ; GET REST OF BAD BLOCKS (15 AND 16)
060007                  T4SOFT  =       7.+DUPPKT       ; ADD TO SOFT ERROR AND ECC COUNT
060010                  T4SEEK  =       8.+DUPPKT       ; ADD 1 TO SEEK COUNT
060011                  T4MXFR  =       9.+DUPPKT       ; ADD TO MEGABITS READ AND WRITTEN
060012                  UTOTST  =       10.+DUPPKT      ; GET UNITS TO TEST
060013                  ERRMES  =       11.+DUPPKT      ; PRINT ERROR MESSAGE
060014                  ERRMC   =       12.+DUPPKT      ; TEST 4 ERROR REPORTING
060015                  MESSAG  =       13.+DUPPKT      ; INFORMATION MESSAGE
060016                  DONE    =       14.+DUPPKT      ; MARK DM PROGRAM AS NO LONGER RUNNING
                                ;
                                ;       STATE BIT DEFINITIIONS
                                ;
000001                  RCVRDY  =       1               ; RECIEVER READY 1 = READY
000002                  ATTN    =       2               ; ATTENTION BIT FOR RETURN DRIVE SIGNALS XFC
000004                  DCLOCK  =       4               ; CONTROLLER STATE READY -- THERE ARE CLOCKS
                                                        ; COMING FROM THE DRIVE
000100                  AVAIL   =       100             ; AVAILABLE 1 = AVAILABLE
000400                  RCVERR  =       400             ; RECEIVE ERROR -- PARITY ERROR
100000                  RWRDY   =       100000          ; IF SET, UDA IS ABLE TO READ AND/OR WRITE TO DRIVE
                                ;
                                ;       SDI COMMANDS AND RESPONSES
                                ;
000204                  DISCON  =       204             ; DISCONNECT DRIVE
000006                  ERECOV  =       6               ; ERROR RECOVERY
000201                  CHGMOD  =       201             ; CHANGE MODE
000213                  DRVONL  =       213             ; DRIVE ONLINE
000014                  DRVRUN  =       14              ; DRIVE RUN
000005                  DRVCLR  =       5               ; DRIVE CLEAR OPCODE
000207                  GETCHR  =       207             ; GET CHARACTERISTICS
000210                  GETSUB  =       210             ; GET SUBUNIT CHARACTERISTICS
000011                  GETSTA  =       11              ; GET STATUS
000216                  IRECLB  =       216             ; RECALIBRATE
000012                  INSEEK  =       12              ; INITIATE SEEK
000176                  COMPLT  =       176             ; SUCCESSFUL COMPLETION
000175                  UNSSUC  =       175             ; UNSUCCESSFUL COMPLETION
000170                  CHRRES  =       170             ; GET CHARACTERISTICS RESPONSE
000167                  SBCRES  =       167             ; GET SUBUNIT CHARACTERISTICS RESPONSE
000366                  STSRES  =       366             ; GET STATUS RESPONSE
000350                  ECHOC   =       350             ; DIAGNOSTIC ECHO COMMAND AND RESPONSE
                                ;
                                ;       SDI LEVEL 2 COMMAND OFFSETS
                                ;
000000                  L2.OPC  =       0               ; SDI LEVEL 2 OP CODE
000001                  L2.SLN  =       L2.OPC+1        ; SDI LEVEL 2 SEND LENGTH IN BYTES
000002                  L2.RLN  =       L2.SLN+1        ; SDI LEVEL 2 RECEIVE LENGTH IN WORDS
000003                  L2.ERS  =       L2.RLN+1        ; SDI LEVEL 2 EXPECTED RESPONSE
000004                  L2.EOF  =       L2.ERS+1        ; SDI LEVEL 2 ERROR OFFSET INTO UNIT PARAMETERS
                                ;
```

```
                                ;       ERROR CODES
                                ;
000000                          FTLSYS  =       0               ; SYSTEM FATAL ERROR
040000                          FTLDEV  =       040000          ; DEVICE FATAL
100000                          ERHARD  =       100000          ; HARD ERROR
140000                          ERSOFT  =       140000          ; SOFT ERROR
040000                          C2HARD  =       <ERHARD&^CERSOFT>!<ERSOFT&^CERHARD>     ; CHANGE SOFT TO HARD ERROR
100000                          C2DFTL  =       <FTLDEV&^CERSOFT>!<ERSOFT&^CFTLDEV>     ; CHANGE SOFT TO DEVICE FATAL
```

TEST 4 SPECIFIC INFORMATION

```
                                .SBTTL  TEST 4 SPECIFIC INFORMATION
                                ;
                                ;       TEST 4 SPECIFIC INFORMATION
                                ;
                                ;
                                ;
                                ;       UNIT PARAMETER OFFSETS
                                ;
                                ;
000000          U.NEXT  =       0.              ; POINTER TO NEXT UNIT (RING LINKED LIST)
000001          U.SUBP  =       U.NEXT+1        ; 4 WORDS OF SUBUNIT PARAMETER POINTERS
000005          U.TIMH  =       U.SUBP+4        ; HI ORDER TIMEOUT FOR SEEK
000006          U.TIML  =       U.TIMH+1        ; LO ORDER TIMEOUT FOR SEEK
000007          U.SRTY  =       U.TIML+1        ; ISSUE SEEK TIMEOUT COUNTER
000010          U.RRTY  =       U.SRTY+1        ; READ RETRIES
000011          U.MSTO  =       U.RRTY+1        ; MASTER SEEK TIMEOUT
000012          U.RWTO  =       U.MSTO+1        ; READ/WRITE TIMEOUT AREA
000013          U.NFUN  =       U.RWTO+1        ; NEXT FUNCTION ADDRESS (FOR DEFERRED CALLS)
000014          U.PAT   =       U.NFUN+1        ; PATTERN NUMBER TO WRITE
000015          U.CCNT  =       U.PAT+1         ; CURRENT COUNT OF T/G LOOPS
000017          U.PCTG  =       U.CCNT+2        ; POINTER TO CURRENT TRACK OR GROUP
000020          U.CTRK  =       U.PCTG+1        ; TRACK COUNT FOR GROUP OPERATIONS
000021          U.NSEC  =       U.CTRK+1        ; NUMBER OF SECTORS R/W THIS TRY
000022          U.MSEC  =       U.NSEC+1        ; NUMBER OF SECTORS TO BE R/W
000023          U.TSEC  =       U.MSEC+1        ; NUMBER OF SECTORS TO BE R/W THIS OP
000024          U.CSEC  =       U.TSEC+1        ; COUNT OF SECTORS R/W SO FAR
000025          U.MASK  =       U.CSEC+1        ; UNIT MASK FOR XFC CALLS (0001 - 1000)
000026          U.WRIT  =       U.MASK+1        ; WRITE PROTECTION STATUS
000027          U.ELEV  =       U.WRIT+1        ; CURRENT ERROR RECOVERY LEVEL
000030          U.RTRY  =       U.ELEV+1        ; MAXIMUM NUMBER OF READ RETRIES
000031          U.MLEV  =       U.RTRY+1        ; MAXIMUM NUMBER OF ERROR RECOVERY LEVELS
000032          U.ECCT  =       U.MLEV+1        ; ECC THRESHOLD
000033          U.SDIS  =       U.ECCT+1        ; SDI SHORT TIMEOUT
000034          U.SDIL  =       U.SDIS+1        ; SDI LONG TIMEOUT
000035          U.SDI2  =       U.SDIL+1        ; LEVEL 2 ERROR COUNTS
000045          U.WPRT  =       U.SDI2+NUML2S   ; MASK TO WRITE PROTECT READ-ONLY DRIVES
000046          U.PARM  =       U.WPRT+1        ; UNIT PARAMETER WORD
000047          U.RCOV  =       U.PARM+1        ; RECOVERY PARAMETERS
000050          U.SUBU  =       U.RCOV+1        ; SUBUNIT OFFSET (0 - 3)
000051          U.MBN   =       U.SUBU+1        ; MASTER L/DBN
000053          U.CBN   =       U.MBN+2         ; CURRENT L/DBN FOR START OF CHAIN
000055          U.RBN   =       U.CBN+2         ; RBN TO BE READ/WRITTEN IF LBN REVECTORED
000057          U.COPY  =       U.RBN+2         ; NUMBER OF RCT COPIES ON EACH SUBUNIT
000060          U.CCOP  =       U.COPY+1        ; CURRENT RCT COPY THAT WE'RE WORKING ON
000061          U.RWER  =       U.CCOP+1        ; ERROR (IF ANY) ON THE LAST R/W
000062          U.RVER  =       U.RWER+1        ; ERROR THAT OCCURRED BEFOR THE REVECTOR OPERATION
000063          U.UNUM  =       U.RVER+1        ; STARTING SUBUNIT NUMBER
000064          U.CCYL  =       U.UNUM+1        ; CURRENT CYLINDER
000066          U.CGRP  =       U.CCYL+2        ; CURRENT GROUP
000067          U.LCYL  =       U.CGRP+1        ; LAST CYLINDER SEEKED TO
000071          U.LGRP  =       U.LCYL+2        ; LAST GROUP SEEKED TO
                                ;
                                ;       SUBUNIT PARAMETER OFFSET
                                ;
000000          S.PARM  =       0.              ; SUBUNIT PARAMETER WORD
000001          S.SEEK  =       S.PARM+1        ; COUNT OF NUMBER OF SEEKS
000002          S.SDCL  =       S.SEEK+1        ; STARTING DIAGNOSTIC CYLINDER
```

```
                                ;
                                ;       IF TRACK/GROUP LIMITS ARE GIVEN, THE SUBUNIT PARAMETERS HAVE THE
                                ;       FOLLOWING FIELDS ADDED TO THEM
                                ;
000013          S.MCNT  =       S.BESS          ; MAXIMUM TRACK/GROUP COUNT
000015          S.TGOF  =       S.MCNT+2        ; ORIGINAL TRACK/GROUP OFFSET
000017          S.TGSS  =       S.TGOF+2        ; START OF TRACK/GROUP SETS
                                ;
                                ;       CONSTANTS
                                ;
000377          SCTWRD  =       255.            ; NUMBER OF WORDS IN SECTOR TO FILL
000105          INTEDC  =       69.             ; INITIAL EDC VALUE
000072          TLEN.U  =       U.LGRP+1        ; UNIT PARAMETER LENGTH
037705          FIRSTU  =       HIMEM-TLEN.U    ; LOCATION OF FIRST UNIT PARAMETER BLOCK
177740          MAXMSK  =       177740          ; TO DETERMINE MAXIMUM SECTORS TO READ/WRITE
                                                ; MAXIMUM SECTOR MASK IS THE 2'S COMPLEMENT
                                                ; OF THE MAXIMUM NUMBER OF SECTORS TO WRITE
001750          MAXSND  =       1000.           ; MAXIMUM TIMES TO SEND A COMMAND TO AN OFFLINE DRIVE
```

```
                                ;
                                ;       DUMMY SDI CONTROL BLOCK OFFSETS
                                ;
000001                          D.LIMT  =       1               ; DUMMY SDI SEARCH LIMIT
000002                          D.SCHR  =       2               ; DUMMY POINTER TO SUBUNIT CHAR-5
                                ;
                                ;       MASTER PARAMETER BITS (M.PARM)
                                ;
100000                          IWIPRG  =       100000          ; INITIAL WRITE IN PROGRESS
000002                          DIE     =       000002          ; ERRORS DURING INITIALIZATION
000001                          INTINP  =       000001          ; INITIAL SETUP IN PROGRESS
                                ;
                                ;       UNIT PARAMETER BITS (U.PARM(R5))
                                ;
100000                          DROP    =       100000          ; DROP BIT (SET IF UNIT OR SUBUNIT DROPPED)
040000                          INITW   =       40000           ; INITIAL WRITE (SET IF INITIAL WRITE IN PROG)
010000                          DIREC   =       10000           ; DIRECTION (SET IF SEQUENTIAL ACCESSES DECREASING)
004000                          NEWSUB  =       4000            ; SET IF SEQUENTIAL SEEKS MOVED TO NEW SUBUNIT
002000                          SEKINP  =       2000            ; SEEK IN PROGRESS - SET IF TRUE
001000                          FTIME   =       1000            ; FIRST TIME FLAG - SET FOR INIT CODE
000400                          REVEC   =       400             ; REVECTOR BIT (SET IF BLOCK REVECTORED)
000200                          RBNBN   =       200             ; SEE IF WORKING ON RBN
000100                          REDWRT  =       100             ; REDWRT (READ OR WRITE IN PROGRESS SET IF WRITE)
                                ;               10              ; RESERVED (SEE SUBUNIT PARAMETER BITS)
                                ;               2               ; RESERVED (SEE SUBUNIT PARAMETER BITS)
                                ;
                                ;       ERROR RECOVERY PARAMETERS (U.RCOV)
                                ;
100000                          ERMASK  =       100000          ; IF 1, ERROR RECOVERY IS DISABLED
040000                          DRINIT  =       40000           ; IF 1, INDICATES THAT DRIVE WAS INITIALIZED
020000                          LEVUSD  =       20000           ; IF 1, RE-ISSUE THIS LEVEL OF ERROR RECOVERY
010000                          NXTLEV  =       10000           ; IF 1, ISSUE NEW LEVEL OF ERROR RECOVERY
004000                          SEKREQ  =       4000            ; IF 1, SEEK IS REQUIRED FOR ERROR RECOVERY
002000                          RCBREQ  =       2000            ; IF 1, RECALIBRATE IS REQUIRED FOR ERROR RECOVERY
001000                          RETRY   =       1000            ; IF 1, JUST RETRY THE SAME MODULE
                                ;
                                ;       SUBUNIT PARAMETER BITS (S.PARM(R4))
                                ;
                                ;               100000          ; RESERVED (DROP BIT)
                                ;               40000           ; RESERVED (INITIAL WRITE)
020000                          DCYLS   =       20000           ; DIAGNOSTIC CYLINDER FLAG (SET IF DBNS)
010000                          ECCCHK  =       10000           ; 1 IF ECC CORRECTION ALLOWED   MASTER BITS
004000                          RONLY   =       4000            ; READ ONLY (SET IF READ ONLY)
002000                          WONLY   =       2000            ; WRITE ONLY (SET IF WRITE ONLY)
001000                          RTRIES  =       1000            ; 1 IF RETRIES ALLOWED
000200                          ONLYCL  =       200             ; SET IF ONLY CYLINDERS SPECIFIED
                                                                ; USED DURING SETUP ONLY
000100                          SEQSEK  =       100             ; SEQUENTIAL SEEK (START UP TESTING ONLY)
000040                          BEUSED  =       40              ; BEGIN/END SETS (USED IF SET)
000020                          TRACKS  =       20              ; TRACKS OR GROUPS (TRACK IF SET)
000010                          WCHECK  =       10              ; WRITE CHECK BIT (IF SET, WRITE CHECK WILL BE DONE)
                                ;                       WCHECK IS ALSO USED FOR THE UNIT PARAMETERS
000004                          WCHKAL  =       4               ; SET IF WRITE CHECK ALWAYS TO BE DONE
000002                          DATCMP  =       2               ; DATA COMPARE (SET IF DATA COMPARE TO BE DONE)
```

```
58
59          000001                  ;
                                    DCMPAL  =       1
                                    DATCMP IS ALSO USED FOR THE UNIT PARAMETERS
                                          ; SET IF DATA COMPARE ALWAYS TO BE DONE
```

```
                        .SBTTL  MACRO DEFINITIONS
                        ;
                        ;       DIAGNOSTIC MACRO FOR TEST4 OVERLAYS
                        ;
                                .MACRO  DIAG$$
                                TST     $$DIAG+$DIAG$
                                BEQ     .+6
                                MOV     #60000,R0
                                MOV     R0,@$$DIAG+$DIAG$
                                .NLIST
                                .IF     NE,UDA50
                                .LIST
                                BR      .+1
                                .NLIST
                                .ENDC
                                .IF     NE,UDA52
                                .LIST
                                .ERROR  ; NOT CHECKED OUT YET
                                BR      .-2
                                .NLIST
                                .ENDC
                                .LIST
                        $DIAG$  =       $DIAG$ + 1
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;       OVERLAY INFORMATION MACRO
                        ;
                                .MACRO  DFOVLY  LABL$,ONAME,SAREA,EAREA
                                .WORD   <SAREA-PART0>/2
                                .IF     NB,EAREA
                                .WORD   <EAREA-SAREA>/2
                                .IFF
                                .WORD   0
                                .ENDC
                                .WORD   0
                                .WORD   OVL.'ONAME'*4
                                .IF     NE,OCNT$-LABL$
                                .ERROR  ; OVERLAY NUMBER AND POSITION IN TABLE DO NOT MATCH
                                .ENDC
                        OCNT$   =       OCNT$+1
                                .ENDM
                        ;
                        ;       MESSAGE CONTROL TABLE MACRO
                        ;
                                .MACRO  MSG CMDBUF,CMDSZ,RPLSZ,SUCCOM
                                .WORD   CMDBUF                  ; ADDRESS OF COMMAND
                                .WORD   CMDSZ                   ; SIZE OF COMMAND IN BYTES
                                .WORD   RPLSZ                   ; SIZE OF REPLY IN WORDS
                                .WORD   SUCCOM                  ; SUCCESSFUL COMPLETION CODE
                                .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
                        SDI$    =       SDI$+1
                                .ENDM
```

MACRO DEFINITIONS

```
      1                                 .MACRO  BCS     LAB..
      2                                 .NLIST
      3                                 .IF     NE,UDA50
      4                                 .LIST
      5                                                         BCC     .+2
      6                                 .NLIST
      7                                 .ENDC
      8                                 .IF     NE,UDA52
      9                                 .LIST
     10                                                         BCC     .+3
     11                                 .NLIST
     12                                 .ENDC
     13                                 .LIST
     14                                                         BR      LAB..
     15                                 .ENDM
```

MACRO DEFINITIONS

```
                        ;       PUSH REGISTER MACRO
                        ;
                                .MACRO PUSH R9
                                .IRP X,<R9>
                                MOV X,-(SP)
                                .ENDR
                                .ENDM
                        ;
                        ;       POP REGISTER MACRO
                        ;
                                .MACRO POP R9
                                .IRP X,<R9>
                                MOV (SP)+,X
                                .ENDR
                                .ENDM
```

```
                        ;
                        ;       PRIMARY ERROR REPORTING (TEST 4)
                        ;

                        .MACRO  SOFTER  NUM,ARGS
                        ERROR   ERSOFT,NUM,<ARGS>
                                                                MOV     #ERRMES,HRQ.RQ
                        .ENDM

                        .MACRO  REPSFT  SFTFLG,ECCFG,SEKFLG,COMFLG
                        .IF     NB,SFTFLG
                        .IF     IDN,SFTFLG,SOFT
                                                                MOV     #1,HRQ.02
                        .IFF
                        .ERROR  ; NON SOFT ERROR FLAG IN SOFT ERROR POSITION
                        .ENDC
                        .IFF
                                                                CLR     HRQ.02
                        .ENDC
                        .IF     NB,ECCFG
                        .IF     IDN,ECCFG,ECC
                                                                MOV     #1,HRQ.03
                        .IFF
                        .ERROR  ; NON ECC ERROR FLAG IN ECC ERROR POSITION
                        .ENDC
                        .IFF
                                                                CLR     HRQ.03
                        .ENDC
                        .IF     NB,SEKFLG
                        .IF     IDN,SEKFLG,SEEK
                                                                MOV     #1,HRQ.04
                        .IFF
                        .ERROR  ; NON SEEK ERROR FLAG IN SEEK ERROR POSITION
                        .ENDC
                        .IFF
                                                                CLR     HRQ.04
                        .ENDC
                ;       .IF     NB,COMFLG
                ;       .IF     IDN,COMFLG,COMM
                ;                                               MOV     #1,HRQ.05
                ;       .IFF
                ;       .ERROR  ; NON COMMUMICATION ERROR FLAG IN COMMUNICATION ERROR POSITION
                ;       .ENDC
                ;       .IFF
                ;                                               CLR     HRQ.05
                ;       .ENDC
                                                                PUSH    R0
                                                                MOV     U.UNUM(R5),R0
                                                                ADD     U.SUBU(R5),R0
                                                                MOV     R0,HRQ.01
                                                                MOV     #T4SOFT,R0
                                                                CALL    HOSTRQ
                                                                POP     R0
                        .ENDM

                        .MACRO  HARDER  NUM,ARGS
                        ERROR   ERHARD,NUM,<ARGS>
                                                                MOV     #ERRMC,HRQ.RQ
```

```
                        .ENDM

                        .MACRO  DEVFTL  NUM,ARGS
                        ERROR   FTLDEV,NUM,<ARGS>
                                                        MOV     #ERRMC,HRQ.RQ
                        .ENDM

                        .MACRO  SYSFTL  NUM,ARGS
                        ERROR   FTLSYS,NUM,<ARGS>
                                                        MOV     #ERRMC,HRQ.RQ
                        .ENDM

                        .MACRO  ERROR   TYPE,NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       5
                        MOVMSG  #ER'NUM,4
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC
                                                        MOV     #NUM!TYPE+4000.,R2
                                                        MOV     R2,HRQ.02
                                                        MOV     #.,HRQ.01
                        .RADIX
                        .ENDM

                        .MACRO  CERROR  STNUM,ARGS
                        .RADIX  10
                NUMPTR  =       STNUM
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .RADIX
                        .ENDM

                        .MACRO  ERRORC  ARGS
                        .RADIX  10
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .RADIX
                        .ENDM

                        .MACRO  MOVMSG  ARG,INDX

                        .IF     LT,INDX-10
                                                        MOV     ARG,HRQ.0'INDX
                        .IFF
                                                        MOV     ARG,HRQ.'INDX
                        .ENDC
                        .ENDM

                        .MACRO  ENDERR  POS
```

```
                        .RADIX  10
                        .IF     NB,POS
                        NDERR   POS
                        .IFF
                        NDERR   \NUMPTR
                        .ENDC
                        .RADIX
                        .ENDM

                        .MACRO  NDERR   POS
                        .IF     NE,POS
                        MOVMSG  #SER22,POS
                                                                MOV     #POS+1,ERRPOS   ; SET THE POSITION
                        .IF     LT,26.-POS
                        .ERROR  ; STATUS AT END OF ERROR MESSAGE WILL OVERFLOW HOST COMMUNICATION BUFFER
                        .ENDC
                        .IFF
                                                                CLR     ERRPOS          ; CLEAR THE POSITION
                        .ENDC
                        .ENDM
                ;
                ;       MESSAGE REPORTING MACRO
                ;
                        .MACRO  MSSG    NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       3
                        MOVMSG  #MS'NUM,2
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC
                                                                PUSH    R0
                                                                MOV     U.UNLM(R5),HRQ.01
                                                                ADD     U.SUBU(R5),HRQ.01
                                                                MOV     #MESSAG,R0
                                                                CALL    HOSTRQ
                                                                POP     R0
                        .RADIX
                        .ENDM
                ;
                ;
```

MACRO DEFINITIONS

```
                        ;ASSUME MACRO
                                .MACRO  ASSUME  P1,P2
                                .IF     NE,P1-P2
                                .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                .ENDC
                                .ENDM
                        ;
                        ;       XOR THE CONTENTS OF TWO REGISTERS
                        ;
                                .MACRO  RXOR    REG1,REG2
                                MOV     REG2,-(SP)      ; SAVE REGISTER REG2
                                BIC     REG1,REG2               ; CLEAR COORESPONDING BITS IN REG2
                                BIC     (SP)+,REG1      ; CLEAR COORESPONDING BITS IN REG1
                                BIS     REG1,REG2               ; OR WHAT'S LEFT
                                .ENDM
                        ;
                        ;       CONVERT .BLKW CALLS TO ACTUAL WORDS GENERATED
                        ;
                                .MACRO  .BLKW   COUNT
                                .NLIST
                                .REPT   COUNT
                                .WORD   0
                                .ENDR
                                .LIST
                                .ENDM
                        ;
                        ;       MASSIVE INLINE CODE MACROS NOT PRINTED
                        ;
```

```
                                .SBTTL  START OF TEST CODE
                                ;THE FOLLOWING IS FOR DEBUG PURPOSES ONLY. CAN BE NOP OR BREAKPOINT.

001364  000000                        .WORD   0               ; RESERVED LOCATION FOR UDA52 DM ODT

                                ;INITIALIZE STACK

001365  104206  001416                MOV     #STACK,SP       ; SET UP STACK POINTER
001367                                BR      START           ; BRANCH OVER SUPPORT CODE
001367  000000  022654                ^O0,START

                                ;STACK AREA

001371                                .BLKW   21.                     ;STACK
001416  123456                STACK:  .WORD   123456                  ;MARKER FOR STACK UNDERFLOW
```

```
                                .SBTTL  RTDS   - REAL TIME DRIVE STATE ROUTINE WITH ERROR REPORTING (TEST 4)
001417                  RTDS:   CALL    RTDSL           ; GET REAL TIME DRIVE STATE
001417  020000  001453          ^O020000,RTDSL
001421  115002                  TST     R2              ; SEE IF ERROR OCCURRED
001422                          BEQ     1$              ; IF NOT, BRANCH
001422  010000  001451          ^O010000,1$
001424                          BMI     2$              ; IF RECEIVE ERROR, BRANCH
001424  070000  001433          ^O070000,2$
001426                          CERROR  4,#ER17         ; REPORT NO CLOCKS ERROR MESSAGE
001426  104200  001604  001601                                  MOV     #ER17,HRQ.04
001431                          BR      3$              ; BRANCH
001431  000000  001436          ^O00,3$
001433                  2$:     CERROR  4,#ER17A        ; REPORT RECEIVE ERRORS ERROR MESSAGE
001433  104200  001634  001601                                  MOV     #ER17A,HRQ.04
001436  104200  044021  001577  3$:     MOV     #17.!FTLDEV+4000,HRQ.02 ; ERROR NUMBER SET
001441  104200  001441  001576          MOV     #.,HRQ.01       ; ADDRESS OF ERROR SET
001444  104200  060014  001575          MOV     #ERRMC,HRQ.RQ   ; DM REQUEST SET
001447                          ENDERR  0
001447  114000  003013                                  CLR     ERRPOS          ; CLEAR THE POSITION
001451                  1$:     RETURN
001451  000000  000000          ^O00,0

                                .SBTTL  RTDSL  - SAME AS RTDS BUT WITHOUT ERROR REPORTING
001453  104652  000025  RTDSL:  MOV     U.MASK(R5),R2   ; MOVE MASK TO R2
001455                          CALL    RDSTAT          ; GET DRIVE STATUS
001455  020000  001470          ^O020000,RDSTAT
001457  103201  077674          BIC     #^C<RCVRDY!ATTN!AVAIL!RWRDY>,R1 ; CLEAR UNUSED BITS
001461  115002                  TST     R2              ; TEST STATUS FOR ERROR
001462                          BEQ     2$              ; IF NO ERROR, BRANCH
001462  010000  001466          ^O010000,2$
001464  104201  177777          MOV     #-1,R1          ; FLAG AS INVALID
001466                  2$:     RETURN
001466  000000  000000          ^O00,0

                                .SBTTL  RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
001470                  RDSTAT:
                                ;
                                ;RETURN DRIVE STATUS
                                ;STATUS RETURNED IN DM REGISTER 1
                                ;
001470                          PUSH    R0              ; SAVE R0
001470  100467                                                  MOV R0,-(SP)
001471  104207  024000          MOV     #24000,R0       ; MOVE 1/2 SECOND COUNT TO R0
001473  060007          1$:     XFC     STATUS          ; GET REAL TIME DRIVE STATE
001474  102201  000004          BIT     #DCLOCK,R1      ; SEE IF DRIVE CLOCK IS PRESENT
001476                          BEQ     3$              ; IF NOT, BRANCH
001476  010000  001514          ^O010000,3$
001500  102201  000400          BIT     #RCVERR,R1      ; SEE IF RECEIVE ERROR
001502                          BEQ     2$              ; IF NONE, BRANCH
001502  010000  001513          ^O010000,2$
001504  117407                  DEC     R0              ; DECREMENT COUNT
001505                          BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
001505  050000  001473          ^O050000,1$
001507  104202  177777          MOV     #177777,R2      ; MARK AS RECEIVE ERROR
001511                          BR      3$              ; BRANCH OVER 'NO ERRORS'
001511  000000  001514          ^O00,3$
001513  114002          2$:     CLR     R2              ; NO ERRORS
```

```
     42 001514                          3$:     POP     R0              ; RESTORE R0
        001514  104267                                                          MOV (SP)+,R0
     43 001515                                  RETURN                  ; RETURN TO CALLING MODULE
        001515  000000  000000              ^O0,0
```

```
      1                                 .SBTTL  HSTTLK - TALK TO THE HOST (INTERRUPT THE HOST BEFORE 3 MIN)
      2 001517                          TLKHST:
      3                                 ;
      4                                 ;       THIS ROUTINE INTERRUPTS THE HOST WITHOUT CONVEYING ANY USEFUL
      5                                 ;       INFORMATION - ALL IT DOES IS SAY 'I'M STILL ALIVE'
      6                                 ;
      7 001517                          ;       REPSFT  ,,,
        001517  114000  001577                                                  CLR     HRQ.02
        001521  114000  001600                                                  CLR     HRQ.03
        001523  114000  001601                                                  CLR     HRQ.04
        001525  100467                                                                  MOV R0,-(SP)
        001526  104657  000063                                                  MOV     U.UNUM(R5),R0
        001530  105657  000050                                                  ADD     U.SUBU(R5),R0
        001532  104070  001576                                                  MOV     R0,HRQ.01
        001534  104207  060007                                                  MOV     #T4SOFT,R0
        001536  020000  001543          ^020000,HOSTRQ
        001540  104267                                                                  MOV (SP)+,R0
      8 001541                          RETURN
        001541  000000  000000          ^00,0
```

```
                                .SBTTL  HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
001543                          HOSTRQ:
                                ;
                                ;SEND REQUEST BUFFER TO HOST AND WAIT FOR RESPONSE.
                                ;CLEAR ARGUMENT AREA OF OUT BUFFER IN PREPARATION
                                ;FOR NEXT HOSTRQ CALL.
                                ;
                                ;INPUTS:
                                ;       R0 - HOST REQUEST NUMBER
                                ;       OUT BUFFER LOADED WITH DATA

001543                          PUSH    <R0,R1,R2>
001543  100467                                                                  MOV R0,-(SP)
001544  100461                                                                  MOV R1,-(SP)
001545  100462                                                                  MOV R2,-(SP)
001546  104070  001575          MOV     R0,HRQ.RQ               ; STORE REQUEST NUMBER IN BUFFER
001550  104207  001575  SNDAGN: MOV     #HRQ.RQ,R0              ; SEND BUFFER TO HOST
001552  104201  000043          MOV     #BUFSIZ,R1              ; SIZE OF SEND BUFFER
001554  060016                  XFC     MRD                     ; SEND BUFFER
001555  115001                  TST     R1                      ; CHECK FOR ERROR
001556                          BNE     SNDAGN                  ; IF ERROR, TRY AGAIN
001556  050000  001550          ^O50000,SNDAGN
001560  104207  001575          MOV     #HRQ.RQ,R0              ; WAIT FOR RESPONSE FROM HOST
001562  104201  000043          MOV     #BUFSIZ,R1              ; SIZE OR RECEIVE BUFFER
001564  060017                  XFC     MWR                     ; RECEIVE BUFFER
001565  104200  177777  001575  MOV     #-1,HRQ.RQ              ; MAKE REQUEST ILLEGAL
001570                          POP     <R2,R1,R0>
001570  104262                                                                  MOV (SP)+,R2
001571  104261                                                                  MOV (SP)+,R1
001572  104267                                                                  MOV (SP)+,R0
001573                          RETURN
001573  000000  000000          ^O0,0
```

```
                                ;STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS

                                ;HRQ BUFFER - DATA TO SEND/RECEIVE TO/FROM HOST

001575  000000  HRQ.RQ: .WORD 0          ;HOST REQUEST CODE
001576  000000  HRQ.01: .WORD 0          ;DATA ARGUMENT  1
001577  000000  HRQ.02: .WORD 0          ;DATA ARGUMENT  2
001600  000000  HRQ.03: .WORD 0          ;DATA ARGUMENT  3
001601  000000  HRQ.04: .WORD 0          ;DATA ARGUMENT  4
001602  000000  HRQ.05: .WORD 0          ;DATA ARGUMENT  5
001603  000000  HRQ.06: .WORD 0          ;DATA ARGUMENT  6
001604  000000  HRQ.07: .WORD 0          ;DATA ARGUMENT  7
001605  000000  HRQ.08: .WORD 0          ;DATA ARGUMENT  8
001606  000000  HRQ.09: .WORD 0          ;DATA ARGUMENT  9
001607  000000  HRQ.10: .WORD 0          ;DATA ARGUMENT 10
001610  000000  HRQ.11: .WORD 0          ;DATA ARGUMENT 11
001611  000000  HRQ.12: .WORD 0          ;DATA ARGUMENT 12
001612  000000  HRQ.13: .WORD 0          ;DATA ARGUMENT 13
001613  000000  HRQ.14: .WORD 0          ;DATA ARGUMENT 14
001614  000000  HRQ.15: .WORD 0          ;DATA ARGUMENT 15
001615  000000  HRQ.16: .WORD 0          ;DATA ARGUMENT 16
001616  000000  HRQ.17: .WORD 0          ;DATA ARGUMENT 17
001617  000000  HRQ.18: .WORD 0          ;DATA ARGUMENT 18
001620  000000  HRQ.19: .WORD 0          ;DATA ARGUMENT 19
001621  000000  HRQ.20: .WORD 0          ;DATA ARGUMENT 20
001622  000000  HRQ.21: .WORD 0          ;DATA ARGUMENT 21
001623  000000  HRQ.22: .WORD 0          ;DATA ARGUMENT 22
001624  000000  HRQ.23: .WORD 0          ;DATA ARGUMENT 23
001625  000000  HRQ.24: .WORD 0          ;DATA ARGUMENT 24
001626  000000  HRQ.25: .WORD 0          ;DATA ARGUMENT 25
001627  000000  HRQ.26: .WORD 0          ;DATA ARGUMENT 26
001630  000000  HRQ.27: .WORD 0          ;DATA ARGUMENT 27
001631  000000  HRQ.28: .WORD 0          ;DATA ARGUMENT 28
001632  000000  HRQ.29: .WORD 0          ;DATA ARGUMENT 29
001633  000000  HRQ.30: .WORD 0          ;DATA ARGUMENT 30
001634  000000  HRQ.31: .WORD 0          ;DATA ARGUMENT 31
001635  000000  HRQ.32: .WORD 0          ;DATA ARGUMENT 32
001636  000000  HRQ.33: .WORD 0          ;DATA ARGUMENT 33
001637  000000  HRQ.34: .WORD 0          ;DATA ARGUMENT 34

        000043  BUFSIZ  =       . - HRQ.RQ          ;SIZE OF BUFFER
```

```
                                        .SBTTL  CMPEDC - EDC CALCULATION ROUTINE
001640                                  CMPEDC:
                                        ;
                                        ;       THIS WILL COMPUTE THE EDC OF A SECTOR POINTED TO BY R0, RETURNING
                                        ;       THE VALUE IN R2
                                        ;
001640                                          PUSH    <R4,R1,R0>      ; SAVE R1 AND R0 AND R4
001640  100464                                                                  MOV R4,-(SP)
001641  100461                                                                  MOV R1,-(SP)
001642  100467                                                                  MOV R0,-(SP)
001643  104201  000400                         MOV     #256.,R1        ; COMPUTE OVER 256 WORDS
001645  104202  000105                         MOV     #INTEDC,R2      ; MOVE INITIAL EDC VALUE TO R2
001647  104274                          EDCLOP: MOV     (R0)+,R4        ; GET A WORD
001650                                          RXOR    R4,R2           ; EXCLUSIVE OR THE WORD WITH THE CURRENT EDC VALUE
001650  100462                                  MOV     R2,-(SP)        ; SAVE REGISTER R2
001651  103042                                  BIC     R4,R2           ; CLEAR COORESPONDING BITS IN R2
001652  103264                                  BIC     (SP)+,R4        ; CLEAR COORESPONDING BITS IN R4
001653  101042                                  BIS     R4,R2           ; OR WHAT'S LEFT
001654  105022                                  ADD     R2,R2           ; SHIFT R2 LEFT BY 1
001655                                          BCC     ZEREDC          ; IF THE HIGH BIT WAS CLEAR, BRANCH
001655  040000  001660                         ^O040000,ZEREDC
001657  115402                                  INC     R2              ; SET LO BIT TO 1 (OLD HI BIT)
001660  117401                          ZEREDC: DEC     R1              ; DECREMENT COUNT
001661                                          BNE     EDCLOP          ; IF COUNT UNEXPIRED, BRANCH
001661  050000  001647                         ^O050000,EDCLOP
001663                                          POP     <R0,R1,R4>      ; RESTORE R1 AND R0 AND R4
001663  104267                                                                  MOV (SP)+,R0
001664  104261                                                                  MOV (SP)+,R1
001665  104264                                                                  MOV (SP)+,R4
001666                                          RETURN
001666  000000  000000                         ^O00,0
```

```
                                .SBTTL  TALK   - SDI LEVEL 2 INTERCHANGE ROUTINE
001670                          TALK:
                                ;
                                ;       TALK SENDS THE COMMAND TO THE DRIVE.  IF AN ERROR OCCURRS, R2 IS
                                ;       RETURNED NONZERO
                                ;
001670                                  PUSH    <R4,R5>         ; SAVE POINTER TO UNIT AND SUBUNIT PARAMETERS
001670  100464                                                                          MOV R4,-(SP)
001671  100465                                                                          MOV R5,-(SP)
001672  104652  000025                  MOV     U.MASK(R5),R2   ; GET UNIT SDI SELECT MASK
001674  104205  000001                  MOV     #1,R5           ; SEND TIMEOUT DEFAULT IS ONE
001676  106203  002356                  CMP     #SNDONE,R3      ; SEE IF ONLY TO BE SENT ONCE
001700                                  BMI     1$              ; IF SO, BRANCH
001700  070000  001704                  ^O070000,1$
001702  104205  001750                  MOV     #MAXSND,R5      ; SEND MAXIMUM NUMBER OF TIMES (ONLINE)
001704  104137                  1$:     MOV     (R3),R0         ; POINTS TO SDI COMMAND BUFFER
001705                                  ASSUME  L2.OPC,0
001705  104631  000001                  MOV     L2.SLN(R3),R1   ; LOAD BYTE COUNT
001707  060004                          XFC     SEND            ; SEND SDI COMMAND
001710  115001                          TSTB    R1              ; SEE IF SDI COMMAND SENT SUCESSFULLY
001711                                  BEQ     2$              ; IF SO, BRANCH
001711  010000  001741                  ^O010000,2$
001713  117405                          DEC     R5              ; DECREMENT TIMEOUT
001714                                  BNE     1$              ; IF UNEXPIRED, BRANCH
001714  050000  001704                  ^O050000,1$
001716                                  POP     R5              ; RESTORE R5
001716  104265                                                                          MOV (SP)+,R5
001717                                  PUSH    R3              ; SAVE SDI PACKET POINTER
001717  100463                                                                          MOV R3,-(SP)
001720                                  CALL    GOINIT          ; INIT THE DRIVE
001720  020000  005106                  ^O020000,GOINIT
001722                                  SOFTER  70              ; REPORT
001722  104200  004261  001601                                                                 MOV     #ER70,HRQ.04
001725  104202  147746                                                                         MOV     #70!ERSOFT+4000.,R2
001727  104020  001577                                                                         MOV     R2,HRQ.02
001731  104200  001731  001576                                                                 MOV     #.,HRQ.01
001734  104200  060013  001575                                                                 MOV     #ERRMES,HRQ.RQ
001737                                  BR      11$             ; BRANCH
001737  000000  002166                  ^O00,11$
001741                          2$:     POP     R5              ; RESTORE R5
001741  104265                                                                          MOV (SP)+,R5
001742  104654  000033                  MOV     U.SDIS(R5),R4   ; R4 HAS SHORT TIMEOUT
001744  106203  002414                  CMP     #LONG,R3        ; SEE IF LONG TIMEOUT TO BE USED
001746                                  BPL     4$              ; IF SO, BRANCH
001746  030000  001752                  ^O030000,4$
001750  104654  000034                  MOV     U.SDIL(R5),R4   ; R4 HAS LONG TIMEOUT
001752                          4$:     PUSH    R3              ; SAVE POINTER
001752  100463                                                                          MOV R3,-(SP)
001753                                  CALL    RTDS            ; GET REAL TIME DRIVE STATUS
001753  020000  001417                  ^O020000,RTDS
001755  115002                          TSTB    R2              ; OK?
001756                                  BEQ     5$              ; IF SO, BRANCH
001756  010000  001764                  ^O010000,5$
001760                                  CALL    GORTRY          ; ELSE, RETRY
001760  020000  005056                  ^O020000,GORTRY
001762                                  BR      17$             ; AND BRANCH DUE TO ERROR
001762  000000  002347                  ^O00,17$
```

```
     37 001764  110601                    5$:     ROR     R1              ; IS RCVRDY SET?
     38 001765                                    BCC     6$              ; IF NOT, BRANCH
        001765  040000  002006                    ^040000,6$
     39 001767                                    ASSUME RCVRDY,1
     40 001767                                    SOFTER  77
        001767  104200  004523  001601                                            MOV     #ER77,HRQ.04
        001772  104202  147755                                                    MOV     #77!ERSOFT+4000.,R2
        001774  104020  001577                                                    MOV     R2,HRQ.02
        001776  104200  001776  001576                                            MOV     #.,HRQ.01
        002001  104200  060013  001575                                            MOV     #ERRMES,HRQ.RQ
     41 002004                                    BR      11$             ; END ERROR
        002004  000000  002166                    ^00,11$
     42 002006  104207  002442          6$:     MOV     #ST,R0          ; POINT TO RECEIVE BUFFER
     43 002010  104631  000002                    MOV     L2.RLN(R3),R1   ; NUMBER OF WORDS IN RESPONSE
     44 002012  104652  000025                    MOV     U.MASK(R5),R2   ; STORE MASK IN R2
     45 002014  060005                            XFC     RCV             ; RECEIVE SDI RESPONSE
     46 002015  115001                            TST     R1              ; SEE IF SDI RESPONSE RECEIVED SUCESSFULLY
     47 002016                                    BEQ     13$             ; IF SO, BRANCH
        002016  010000  002225                    ^010000,13$
     48 002020  106201  000001                    CMP     #1,R1           ; SEE IF TIMEOUT
     49 002022                                    BNE     7$              ; IF NOT, BRANCH
        002022  050000  002054                    ^050000,7$
     50 002024                                    CALL    TLKHST          ; SAY 'I'M ALIVE' TO THE HOST
        002024  020000  001517                    ^020000,TLKHST
     51 002026                                    POP     R3              ; RESTORE R3
        002026  104263                                                              MOV (SP)+,R3
     52 002027  117404                            DEC     R4              ; DECREMENT TIMEOUT VALUE
     53 002030                                    BNE     4$              ; IF TIMEOUT UNEXPIRED, BRANCH
        002030  050000  001752                    ^050000,4$
     54 002032                                    PUSH    R3              ; SAVE R3
        002032  100463                                                              MOV R3,-(SP)
     55 002033                                    CALL    GOINIT          ; INIT THE DRIVE
        002033  020000  005106                    ^020000,GOINIT
     56 002035                                    SOFTER  71              ; FLAG AS ERROR
        002035  104200  004275  001601                                            MOV     #ER71,HRQ.04
        002040  104202  147747                                                    MOV     #71!ERSOFT+4000.,R2
        002042  104020  001577                                                    MOV     R2,HRQ.02
        002044  104200  002044  001576                                            MOV     #.,HRQ.01
        002047  104200  060013  001575                                            MOV     #ERRMES,HRQ.RQ
     57 002052                                    BR      11$             ; BRANCH
        002052  000000  002166                    ^00,11$
     58 002054  106201  000002          7$:     CMP     #2,R1           ; SEE IF FIRST WORD NOT START FRAME
     59 002056                                    BNE     8$              ; IF NOT, BRANCH
        002056  050000  002077                    ^050000,8$
     60 002060                                    SOFTER  72              ; REPORT FIRST WORD NOT START FRAME
        002060  104200  004312  001601                                            MOV     #ER72,HRQ.04
        002063  104202  147750                                                    MOV     #72!ERSOFT+4000.,R2
        002065  104020  001577                                                    MOV     R2,HRQ.02
        002067  104200  002067  001576                                            MOV     #.,HRQ.01
        002072  104200  060013  001575                                            MOV     #ERRMES,HRQ.RQ
     61 002075                                    BR      11$             ; BRANCH
        002075  000000  002166                    ^00,11$
     62 002077  106201  000004          8$:     CMP     #4,R1           ; SEE IF FRAMING ERROR
     63 002101                                    BNE     9$              ; IF NOT, BRANCH
        002101  050000  002122                    ^050000,9$
     64 002103                                    SOFTER  73              ; REPORT FRAMING ERROR
        002103  104200  004342  001601                                            MOV     #ER73,HRQ.04
```

```
        002106  104202  147751                                                  MOV     #73!ERSOFT+4000.,R2
        002110  104020  001577                                                  MOV     R2,HRQ.02
        002112  104200  002112  001576                                          MOV     #.,HRQ.01
        002115  104200  060013  001575                                          MOV     #ERRMES,HRQ.RQ
    65  002120                                BR      11$             ; BRANCH
        002120  000000  002166                ^O00,11$
    66  002122  106201  000010        9$:     CMP     #10,R1          ; SEE IF CHECKSUM ERROR
    67  002124                                BNE     10$             ; IF NOT, BRANCH
        002124  050000  002145                ^O50000,10$
    68  002126                                SOFTER  74              ; REPORT CHECKSUM ERROR
        002126  104200  004366  001601                                          MOV     #ER74,HRQ.04
        002131  104202  147752                                                  MOV     #74!ERSOFT+4000.,R2
        002133  104020  001577                                                  MOV     R2,HRQ.02
        002135  104200  002135  001576                                          MOV     #.,HRQ.01
        002140  104200  060013  001575                                          MOV     #ERRMES,HRQ.RQ
    69  002143                                BR      11$             ; BRANCH
        002143  000000  002166                ^O00,11$
    70  002145  106201  000020        10$:    CMP     #20,R1          ; SEE IF BUFFER TOO SMALL
    71  002147                                BNE     12$             ; IF NOT, BRANCH
        002147  050000  002176                ^O50000,12$
    72  002151                                SOFTER  75              ; REPORT BUFFER TO SMALL
        002151  104200  004413  001601                                          MOV     #ER75,HRQ.04
        002154  104202  147753                                                  MOV     #75!ERSOFT+4000.,R2
        002156  104020  001577                                                  MOV     R2,HRQ.02
        002160  104200  002160  001576                                          MOV     #.,HRQ.01
        002163  104200  060013  001575                                          MOV     #ERRMES,HRQ.RQ
    73  002166                        11$:    ENDERR  6               ; FLAG END OF REPORTING BUFFER
        002166  104200  000051  001603                                          MOV     #SER22,HRQ.06
        002171  104200  000007  003013                                          MOV     #6+1,ERRPOS     ; SET THE POSITION
    74  002174                                BR      14$             ; EXIT
        002174  000000  002271                ^O00,14$
    75  002176                        12$:    HARDER  78              ; UNKNOWN ERROR RETURNED BY UDA
        002176  104200  004557  001601                                          MOV     #ER78,HRQ.04
        002201  104202  107756                                                  MOV     #78!ERHARD+4000.,R2
        002203  104020  001577                                                  MOV     R2,HRQ.02
        002205  104200  002205  001576                                          MOV     #.,HRQ.01
        002210  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
    76  002213                                CERROR  6,R1            ; REPORT ERROR
        002213  104010  001603                                                  MOV     R1,HRQ.06
    77  002215                                ENDERR  7               ; FLAG END OF REPORTING BUFFER
        002215  104200  000051  001604                                          MOV     #SER22,HRQ.07
        002220  104200  000010  003013                                          MOV     #7+1,ERRPOS     ; SET THE POSITION
    78  002223                                BR      14$             ; EXIT
        002223  000000  002271                ^O00,14$
    79  002225  114002                13$:    CLR     R2              ; FLAG AS NO ERROR OCCURED
    80  002226  106637  000003                CMP     L2.ERS(R3),R0   ; SEE IF COMMAND ACCEPTED
    81  002230                                BEQ     16$             ; IF SO, BRANCH
        002230  010000  002314                ^O10000,16$
    82  002232                                SOFTER  76              ; REPORT
        002232  104200  004444  001601                                          MOV     #ER76,HRQ.04
        002235  104202  147754                                                  MOV     #76!ERSOFT+4000.,R2
        002237  104020  001577                                                  MOV     R2,HRQ.02
        002241  104200  002241  001576                                          MOV     #.,HRQ.01
        002244  104200  060013  001575                                          MOV     #ERRMES,HRQ.RQ
    83  002247                                CERROR  6,<L2.ERS(R3),R0,#SER22> ; REPORT FURTHER ERRORS
        002247  104630  000003  001603                                          MOV     L2.ERS(R3),HRQ.06
        002252  104070  001604                                                  MOV     R0,HRQ.07
```

```
                                                                                    MOV     #SER22,HRQ.08
     84 002257  104200  100011  003013          MOV     #<VALID+11>,ERRPOS ; FLAG AS STATUS GOOD
     85 002262  106207  000175                  CMP     #UNSSUC,R0      ; SEE IF UNSUCCESSFUL RESPONSE
     86 002264                                  BEQ     14$             ; IF SO, BRANCH
        002264  010000  002271                  ^O010000,14$
     87 002266  103200  100000  003013          BIC     #VALID,ERRPOS   ; GET STATUS FOR PRINTING
     88 002271                          14$:    CALL    GORTRY          ; RETRY THIS COMMAND (MINIMUM)
        002271  020000  005056                  ^O020000,GORTRY
     89 002273                                  POP     R3              ; RESTORE POINTER TO COMMAND
        002273  104263                                                                  MOV (SP)+,R3
     90 002274  104633  000004                  MOV     L2.EOF(R3),R3   ; GET SDI ERROR OFFSET
     91 002276  105053                          ADD     R5,R3           ; POINT TO ERROR WORD
     92 002277  104134                          MOV     (R3),R4         ; GET ERROR COUNT
     93 002300  115404                          INC     R4              ; INCREMENT COUNT
     94 002301  106204  000002                  CMP     #2,R4           ; SEE IF MAX
     95 002303                                  BCC     15$             ; IF NOT, BRANCH
        002303  040000  002311                  ^O040000,15$
     96 002305  103200  100000  001577          BIC     #C2DFTL,HRQ.02  ; CHANGE ERROR TO A DEVICE FATAL
     97 002310  114004                          CLR     R4              ; IN CASE DROPS DISABLED, CLEAR ERROR COUNT
     98 002311  100134                  15$:    MOV     R4,(R3)         ; SAVE COUNT
     99 002312                                  BR      17$             ; EXIT
        002312  000000  002347                  ^O00,17$
    100 002314                          16$:    POP     R3              ; RESTORE POINTER TO COMMAND
        002314  104263                                                                  MOV (SP)+,R3
    101 002315  104633  000004                  MOV     L2.EOF(R3),R3   ; GET SDI ERROR OFFSET
    102 002317  105053                          ADD     R5,R3           ; POINT TO ERROR WORD
    103 002320  104134                          MOV     (R3),R4         ; GET ERROR COUNT
    104 002321                                  BEQ     17$             ; IF NO RETRIES, BRANCH
        002321  010000  002347                  ^O010000,17$
    105 002323  100132                          MOV     R2,(R3)         ; ZERO RETRIES
    106 002324                                  REPSFT  SOFT,,,COMM     ; REPORT SOFT AND COMMUNICATION ERRORS
        002324  104200  000001  001577                                         MOV     #1,HRQ.02
        002327  114000  001600                                                  CLR     HRQ.03
        002331  114000  001601                                                  CLR     HRQ.04
        002333  100467                                                                  MOV R0,-(SP)
        002334  104657  000063                                                  MOV     U.UNUM(R5),R0
        002336  105657  000050                                                  ADD     U.SUBU(R5),R0
        002340  104070  001576                                                  MOV     R0,HRQ.01
        002342  104207  060007                                                  MOV     #T4SOFT,R0
        002344  020000  001543                  ^O020000,HOSTRQ
        002346  104267                                                                  MOV (SP)+,R0
    107 002347                          17$:    POP     R4              ; RESTORE R4
        002347  104264                                                                  MOV (SP)+,R4
    108 002350                                  RETURN
        002350  000000  000000                  ^O00,0
```

```
                                .SBTTL  SDI PROTOCOL MESSAGE TABLES AND RESPONSE BUFFER
                                ;
                                ;       MESSAGE TABLES
                                ;
            000000              SDI$    =       0               ; INITIAL OFFSET
002352                          CR.ONL: MSG ONL,2,7,COMPLT      ; BRING DRIVE ONLINE
002352  002440                          .WORD   ONL                     ; ADDRESS OF COMMAND
002353  000002                          .WORD   2                       ; SIZE OF COMMAND IN BYTES
002354  000007                          .WORD   7                       ; SIZE OF REPLY IN WORDS
002355  000176                          .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
002356  000035                          .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
            002356              SNDONE  =       .-1     ; PREVIOUS COMMANDS CAN BE SENT TO AN OFFLINE DRIVE
002357                          CR.GST: MSG GST,1,7,STSRES      ; GET STATUS
002357  002430                          .WORD   GST                     ; ADDRESS OF COMMAND
002360  000001                          .WORD   1                       ; SIZE OF COMMAND IN BYTES
002361  000007                          .WORD   7                       ; SIZE OF REPLY IN WORDS
002362  000366                          .WORD   STSRES                  ; SUCCESSFUL COMPLETION CODE
002363  000036                          .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
002364                          CR.CLR: MSG DRC,2,7,COMPLT      ; DRIVE CLEAR
002364  002426                          .WORD   DRC                     ; ADDRESS OF COMMAND
002365  000002                          .WORD   2                       ; SIZE OF COMMAND IN BYTES
002366  000007                          .WORD   7                       ; SIZE OF REPLY IN WORDS
002367  000176                          .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
002370  000037                          .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
002371                          CR.MOD: MSG MOD,3,7,COMPLT      ; CHANGE MODE
002371  002424                          .WORD   MOD                     ; ADDRESS OF COMMAND
002372  000003                          .WORD   3                       ; SIZE OF COMMAND IN BYTES
002373  000007                          .WORD   7                       ; SIZE OF REPLY IN WORDS
002374  000176                          .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
002375  000040                          .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
002376                          CR.SEK: MSG INS,6,7,COMPLT      ; INITIATE SEEK
002376  002432                          .WORD   INS                     ; ADDRESS OF COMMAND
002377  000006                          .WORD   6                       ; SIZE OF COMMAND IN BYTES
002400  000007                          .WORD   7                       ; SIZE OF REPLY IN WORDS
002401  000176                          .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
002402  000041                          .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
002403                          CR.ERR: MSG ERR,2,7,COMPLT      ; ERROR RECOVERY
002403  002436                          .WORD   ERR                     ; ADDRESS OF COMMAND
002404  000002                          .WORD   2                       ; SIZE OF COMMAND IN BYTES
002405  000007                          .WORD   7                       ; SIZE OF REPLY IN WORDS
002406  000176                          .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
002407  000042                          .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
002410                          CR.DIS: MSG DIS,2,7,COMPLT      ; DISCONNECT DRIVE
002410  002422                          .WORD   DIS                     ; ADDRESS OF COMMAND
002411  000002                          .WORD   2                       ; SIZE OF COMMAND IN BYTES
002412  000007                          .WORD   7                       ; SIZE OF REPLY IN WORDS
002413  000176                          .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
002414  000043                          .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
            002414              LONG    =       .-1     ; ALL COMMANDS BEYOND THIS POINT ARE LONG TIMEOUT
002415                          CR.INR: MSG INR,1,7,COMPLT      ; INITIATE RECALIBRATE
002415  002431                          .WORD   INR                     ; ADDRESS OF COMMAND
002416  000001                          .WORD   1                       ; SIZE OF COMMAND IN BYTES
002417  000007                          .WORD   7                       ; SIZE OF REPLY IN WORDS
002420  000176                          .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
002421  000044                          .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
            000010              NUML2S  =       SDI$            ; SAVE NUMBER OF LEVEL 2 SDI COMMANDS
                                ;
```

```
                                        ;       LEVEL 2 COMMAND MESSAGE DATA STRUCTURES
                                        ;
002422      000         204         DIS:    .BYTE   0,DISCON        ; DISCONNECT DRIVE
002423  000000                              .WORD   0               ; DO NOT SPIN DOWN DRIVE
002424      000         201         MOD:    .BYTE   0,CHGMOD        ; CHANGE MODE
002425  000000                      WRITBT: .WORD   0               ; MODE
002426      000         005         DRC:    .BYTE   0,DRVCLR        ; DRIVE CLEAR
002427  000000                      DCLR:   .WORD   0
002430      000         011         GST:    .BYTE   0,GETSTA        ; GET STATUS
002431      000         216         INR:    .BYTE   0,IRECLB        ; INITIATE RECALIBRATE
002432      000         012         INS:    .BYTE   0,INSEEK        ; INITIATE SEEK
002433  000000                      LOCYL:  .WORD   0               ; INS CYLINDER/HEAD ARGUMENTS
002434  000000                              .WORD   0               ; HI CYLINDER
002435  000000                              .WORD   0               ; GROUP
002436      000         006         ERR:    .BYTE   0,ERECOV        ; ERROR RECOVERY
002437  000000                      ERRLEV: .WORD   0
002440      000         213         ONL:    .BYTE   0,DRVONL        ; ONLINE COMMAND
002441  000012                              .WORD   10.             ; TIMEOUT VALUE


002442                              ST:     .BLKW   19.             ; RESPONSE BUFFER
```

```
                                .SBTTL  BLKCHK - SEE IF A BLOCK WITH ERROR IS KNOWN TO BE BAD
002465                          BLKCHK:
                                ;
                                ;       BLKCHK CHECKS THE BLOCK IN U.LBN TO ASSURE THAT IT IS NOT A BAD
                                ;       BLOCK.  IF SO, R2 IS RETURNED NONZERO
                                ;
002465                          PUSH    <R2,R1,R0>      ; SAVE ALL REG'S
002465  100462                                                          MOV R2,-(SP)
002466  100461                                                          MOV R1,-(SP)
002467  100467                                                          MOV R0,-(SP)
002470  104651  000046          MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
002472  102201  000200          BIT     #RBNBN,R1       ; SEE IF ON RBN
002474                          BNE     1$              ; IF SO, UNKNOWN BAD BLOCK
002474  050000  002523          ^050000,1$
002476  104641  000012          MOV     S.BADP(R4),R1   ; GET BAD BLOCK POINTER
002500                          BEQ     1$              ; IF NO BAD BLOCKS, BRANCH
002500  010000  002523          ^010000,1$
002502  104202  000003          MOV     #RW.LOW,R2      ; POINT TO LBN TO BE TESTED
002504  105072                  ADD     R0,R2           ; POINT TO LBN TO BE TESTED
002505                  3$:     CALL    CMP2            ; COMPARE BAD BLOCK TO LBN
002505  020000  002536          ^020000,CMP2
002507                          BEQ     2$              ; IF EQUAL, BRANCH
002507  010000  002527          ^010000,2$
002511                          BCS     1$              ; IF LIST GREATER THAN BLOCK, IT'S OK
002511  040000  002515          ^040000,.+3
002513  000000  002523          ^00,1$
002515  105201  000002          ADD     #2,R1           ; POINT TO NEXT BAD BLOCK
002517  104617  177777          MOV     -1(R1),R0       ; SEE IF EOL
002521                          BPL     3$              ; IF NOT, BRANCH
002521  030000  002505          ^030000,3$
002523  104207  000001  1$:     MOV     #1,R0           ; SET UP FOR CARRY TO BE SET (UNKNOWN BAD BLOCK)
002525                          BR      4$
002525  000000  002530          ^00,4$
002527  114007          2$:     CLR     R0              ; SET UP FOR THE CARRY TO BE CLEAR (BAD BLOCK KNOWN)
002530  110607          4$:     RORB    R0              ; SET CARRY TO REFLECT KNOWLEDGE OF BAD BLOCK
002531                          POP     <R0,R1,R2>      ; RESTORE
002531  104267                                                          MOV (SP)+,R0
002532  104261                                                          MOV (SP)+,R1
002533  104262                                                          MOV (SP)+,R2
002534                          RETURN                  ; RETURN TO CALLING PROGRAM
002534  000000  000000          ^00,0
```

```
                                        .SBTTL  CMP2    - 28 BIT COMPARE
002536                                  CMP2:
                                        ;
                                        ;       CMP2 COMPARES A 28 BIT NUMBER POINTED TO BY R2 TO A 28 BIT NUMBER
                                        ;       POINTED TO BY R1.  BOTH NUMBERS ARE LOW ORDER WORD FOLLOWED BY HIGH
                                        ;       WORD (POINTER TO LOW WORD) AND THE HIGH 4 BITS <31:28> OF THE
                                        ;       WORD POINTED TO BY R1 ARE STRIPPED OFF (THIS IS TO ELIMINATE THE
                                        ;       END-OF-LIST FLAG ON THE B/E SETS AND BAD BLOCKS)
                                        ;
002536  104617  000001                          MOV     1(R1),R0        ; MOVE HIGH ORDER WORD TO R0
002540  103207  170000                          BIC     #^CHBHINB,R0    ; CLEAR UNUSED BITS
002542  106627  000001                          CMP     1(R2),R0        ; COMPARE HIGH ORDER WORRD TO R0
002544                                          BNE     1$              ; IF UNEQUAL, BRANCH
002544  050000  002550                          ^050000,1$
002546  104117                                  MOV     (R1),R0         ; MOVE LOW ORDER WORD TO R0
002547  106127                                  CMP     (R2),R0         ; COMPARE LOW ORDER WORD TO R0
002550                                  1$:     RETURN                  ; RETURN TO CALLING PROGRAM
002550  000000  000000                          ^00,0
```

```
      1                                     .SBTTL  BULDUM - BUILD THE DUMMY SDI CONTROL BLOCK FOR READS AND WRITES
      2 002552                              BULDUM:
      3                                     ;
      4                                     ;       BULDUM WILL BUILD THE DUMMY SDI CONTROL BLOCK
      5                                     ;
      6 002552  114002                              CLR     R2                      ; NO RBN'S
      7 002553  104643  000000                   MOV     S.PARM(R4),R3   ; GET SUBUNIT PARAMETERS
      8 002555  102203  020000                   BIT     #DCYLS,R3               ; SEE IF IN DIAGNOSTIC AREA
      9 002557                                  BNE     1$                      ; IF SO, BRANCH
        002557  050000  002567                  ^O50000,1$
     10 002561  104642  000007                   MOV     S.SCHR(R4),R2   ; GET SUBUNIT CHARACTERISTICS
     11 002563  104622  000004                   MOV     RBNTRK(R2),R2   ; GET RBN'S/TRACK
     12 002565  103202  177600                   BIC     #HIBYTE!200,R2  ; CLEAR UNUSED BITS
     13 002567  105642  000006           1$:     ADD     S.TRKL(R4),R2   ; GET SECTORS/TRACK
     14 002571  105022                          ADD     R2,R2                   ; DOUBLE SECTORS/TRACK FOR HEADER COMPARE LIMIT
     15 002572  104020  002773                   MOV     R2,DUMSDI+D.LIMT                ; MOVE TO DUMMY SEARCH LIMIT
     16 002574  104643  000007                   MOV     S.SCHR(R4),R3   ; R3 POINTS TO SUBUNIT CHARACTERISTICS
     17 002576  107203  000005                   SUB     #5,R3                   ; R3 POINTS TO SUB CHAR - 5
     18 002600  104030  002774                   MOV     R3,DUMSDI+D.SCHR                ; MOVE TO DUMMY SUB CHAR POINTER
     19 002602                                  RETURN                          ; RETURN TO CALLING PROGRAM
        002602  000000  000000                  ^O0,0
```

```
                                                .SBTTL  CALC    - CALCULATE THE CYL, GRP AND TRACK FOR THE GIVEN L/RBN
                                                CALC:
                                                ;
                                                ;       CALC CALCULATES THE CYLINDER, GROUP, TRACK AND SECTOR OF THE BLOCK
                                                ;
 002604                                         
                                                
                                                
                                                
 002604  115007                                        TST     R0              ; SEE IF CALCULATION AREA MUST BE SET UP
 002605                                                 BNE     MOVOUT          ; IF NOT, BRANCH
 002605  050000  002746                                 ^050000,MOVOUT
 002607  104070  002763                                 MOV     R0,RBNFLG       ; SET RBN FLAG AS ZERO
 002611  104643  000007                                 MOV     S.SCHR(R4),R3   ; R3 POINTS TO SUBUNIT'S CHARACTERISTICS
 002613  104637  000004                                 MOV     RBNTRK(R3),R0   ; GET RBN'S PER TRACK
 002615  103207  177600                                 BIC     #HIBYTE!200,R0  ; CLEAR UNUSED BITS
 002617  105637  000011                                 ADD     LBNTRK(R3),R0   ; ADD LBN'S PER TRACK
 002621  103207  177400                                 BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
 002623  104070  003002                                 MOV     R0,SCR1         ; SAVE
 002625  104147                                        MOV     (R4),R0         ; GET SUBUNIT PARAMETERS
 002626                                                 ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
 002626  102207  020000                                 BIT     #DCYLS,R0       ; SEE IF USING THE DIAGNOSTIC CYLINDERS
 002630                                                 BNE     MOVDBN          ; IF SO, BRANCH
 002630  050000  002705                                 ^050000,MOVDBN
 002632  104637  000011                                 MOV     LBNTRK(R3),R0   ; MOVE LBN'S PER TRACK TO R0
 002634  103207  177400                                 BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
 002636  104651  000046                                 MOV     U.PARM(R5),R1   ; MOVE UNIT PARAMETERS TO R0
 002640  102201  000200                                 BIT     #RBNBN,R1       ; SEE IF BLOCK REVECTORED
 002642                                                 BEQ     LNCYL           ; IF NOT, BRANCH
 002642  010000  002662                                 ^010000,LNCYL
 002644  104070  002763                                 MOV     R0,RBNFLG       ; STORE LBN'S PER TRACK IN RBN FLAG AREA
 002646  104637  000004                                 MOV     RBNTRK(R3),R0   ; MOVE RBN'S PER TRACK TO R0
 002650  103207  177600                                 BIC     #177600,R0      ; CLEAR THE HIGHER BITS, LEAVING RBN'S/TRACK
 002652  104070  002762                                 MOV     R0,LRDTRK       ; MOVE RBN'S PER TRACK TO CALCULATION AREA
 002654  104637  000003                                 MOV     HIRBN(R3),R0    ; GET HI RBN
 002656  103207  170377                                 BIC     #HBLONB,R0      ; CLEAR UNUSED BITS
 002660                                                 BR      LNCONT          ; BRANCH
 002660  000000  002670                                 ^O0,LNCONT
 002662  104070  002762                        LNCYL:   MOV     R0,LRDTRK       ; MOVE LBN'S PER TRACK TO CALCULATION AREA
 002664  104637  000002                                 MOV     HILBN(R3),R0    ; GET HI LBN
 002666  103207  170377                                 BIC     #HBLONB,R0      ; CLEAR UNUSED BITS
 002670  104070  002755                        LNCONT:  MOV     R0,HIBN         ; SAVE
 002672  114007                                         CLR     R0              ; LOW ORDER CYLINDER IS ZERO
 002673  104070  002756                                 MOV     R0,STACYL       ; SAVE LO ORDER STARTING CYLINDER
 002675  104637  000001                                 MOV     HICYL(R3),R0    ; R0 CONTAINS HI CYLINDER BITS
 002677  103207  007777                                 BIC     #HBHINB,R0      ; CLEAR UNUSED BITS
 002701  104070  002757                                 MOV     R0,STACYL+1     ; MOVE HI STARTING CYLINDER TO UNIT PARAMETERS
 002703                                                 BR      MOVOUT          ; BRANCH
 002703  000000  002746                                 ^O0,MOVOUT
 002705  104637  000004                        MOVDBN:  MOV     RBNTRK(R3),R0   ; MOVE NUMBER OF RBN'S PER TRACK TO R0
 002707  103207  177600                                 BIC     #177600,R0      ; CLEAR THE HIGHER BITS, LEAVING RBN'S/TRACK
 002711  104631  000011                                 MOV     LBNTRK(R3),R1   ; ADD LBN'S/TRK, GIVING DBN'S/TRK
 002713  103201  177400                                 BIC     #HIBYTE,R1      ; CLEAR UNUSED BITS
 002715  105017                                         ADD     R1,R0           ; ADD LBNS/TRK TO RBNS/TRK TO GIVE DBNS/TRK
 002716  104070  002762                                 MOV     R0,LRDTRK       ; SAVE DBNS/TRK IN COMPUTATIONAL AREA
 002720  104647  000002                                 MOV     S.SDCL(R4),R0   ; GET LO ORDER STARTING DIAGNOSTIC CYLINDER
 002722  104070  002756                                 MOV     R0,STACYL       ; SAVE LO ORDER CYL
 002724  104637  000001                                 MOV     HICYL(R3),R0    ; GET HI CYLINDER BITS
 002726  103207  007777                                 BIC     #HBHINB,R0      ; CLEAR UNUSED BITS
 002730  101647  000003                                 BIS     S.SDCL+1(R4),R0 ; GET HI ORDER STARTING DIAGNOSTIC CYLINDER
 002732  104070  002757                                 MOV     R0,STACYL+1     ; SAVE HI ORDER CYL
```

```
     53 002734  104637  000003                  MOV     HIDBN(R3),R0    ; GET HI DBN BITS
     54 002736  110607                          ROR     R0              ; ROTATE TO CORRECT POSITION
     55 002737  110607                          ROR     R0              ; ROTATE TO CORRECT POSITION
     56 002740  110607                          ROR     R0              ; ROTATE TO CORRECT POSITION
     57 002741  110607                          ROR     R0              ; ROTATE TO CORRECT POSITION
     58 002742  103207  170377                  BIC     #HBLONB,R0      ; CLEAR UNUSED BITS
     59 002744  104070  002755                  MOV     R0,HIBN         ; SAVE
     60 002746  104207  002756          MOVOUT: MOV     #CALCSC,R0      ; POINT TO CALCULATION AREA
     61 002750  104641  000007                  MOV     S.SCHR(R4),R1   ; POINT TO SUBUNIT CHAR TABLE
     62 002752  060020                          XFC     CVT             ; CALCULATE
     63 002753                                  RETURN                  ; RETURN TO CALLING PROGRAM
        002753  000000  000000                  ^O0,0
```

```
                                .SBTTL  TEST 4 STORAGE AREA FOR VARIABLES
002755                          HIBN:   .BLKW   1                       ; HI BN BITS <27:24>
002756                          CALCSC:                                 ; CALCULATION AREA FOR CYL, GRP, TRK
002756                          STACYL: .BLKW   2                       ; STARTING CYLINDER NUMBER
002760                          CURBN:  .BLKW   2                       ; L/R/DBN
002762                          LRDTRK: .BLKW   1                       ; SECTORS PER TRACK
002763                          RBNFLG: .BLKW   1                       ; RBN FLAG
002764                          CYL:    .BLKW   2                       ; CYLINDER
002766                          GROUP:  .BLKW   1                       ; GROUP
002767                          TRACK:  .BLKW   1                       ; TRACK
002770                          SECTOR: .BLKW   1                       ; SECTOR
002771                          INDEX:  .BLKW   1                       ; INDEX

002772                          DUMSDI: .BLKW   8.                      ; DUMMY SDI AREA

003002                          SCR1:   .BLKW   1.                      ; XFC READ AND WRITE WILL WRITE THE
003003                          SCR2:   .BLKW   1.                      ; REVECTORED LBN IN THIS SPACE,
                                                                        ; OTHERWISE, USE IT FOR SCRATCH

003004                          PNUM:   .BLKW   1                       ; PATTERN NUMBER STORAGE AREA FOR ERRORS
003005                          LSTOVL: .BLKW   1                       ; LAST OVERLAY CALLED NUMBER
003006                          M.PARM. .BLKW   1                       ; MASTER PARAMETERS
003007  034245                  LOSEED: .WORD   34245                   ; LO ORDER RANDOM NUMBER SEED
003010  061453                  HISEED: .WORD   61453                   ; HI ORDER RANDOM NUMBER SEED
003011                          ERMODE: .BLKW   1                       ; UNEXPECTED ATTENTION FLAG
003012  037777                  MEMPOL: .WORD   HIMEM                   ; MEMORY POOL FOR UNIT DATASTRUCTURES
003013                          ERRPOS: .BLKW   1                       ; RTDS AND STATUS POSITION
003014                                  ASSUME .-DUMSDI,18.
                                ; *** NEXT WORD REQUIRED BY UDA TO BE 18 PAST DUMMY SDI BLOCK START
003014  002766                          .WORD   DUMSDI-4                ; POINTER FOR LBN REVECTOR INFORMATION
                                        ;       BELEIVE IT OR NOT, THIS WILL WRITE THE REVECTORED LBN
                                        ;       IN LOCATIONS DUMSDI+8 AND DUMSDI+9
                                        ;       EG. DUMSDI+18. POINTS TO 12 SHORT OF WHERE TO PUT THE
                                        ;             REVECTOR INFORMATION


003015                          CHAINS: .BLKW   1                       ; READ/WRITE CHAIN HEADER
003016                          SECMAX: .BLKW   1                       ; MAXIMUM NUMBER OF BUFFERS THAT CAN BE READ
003017                          CHNMAX: .BLKW   1                       ; MAXIMUM NUMBER OF BUFFERS THAT CAN BE WRITTEN
003020                          MAXSEC: .BLKW   1                       ; MAXIMUM NUMBER OF SECTORS THIS PASS
```

DATA PATTERNS THAT WILL BE WRITTEN TO THE DRIVE

```
                                .SBTTL  DATA PATTERNS THAT WILL BE WRITTEN TO THE DRIVE
003021  003041  PATPTR: .WORD   PAT0                    ; POINTERS TO DATA PATTERNS
003022  003063          .WORD   PAT1
003023  003066          .WORD   PAT2
003024  003071          .WORD   PAT3
003025  003074          .WORD   PAT4
003026  003116          .WORD   PAT5
003027  003140          .WORD   PAT6
003030  003162          .WORD   PAT7
003031  003165          .WORD   PAT8
003032  003207          .WORD   PAT9
003033  003212          .WORD   PAT10
003034  003234          .WORD   PAT11
003035  003237          .WORD   PAT12
003036  003261          .WORD   PAT13
003037  003303          .WORD   PAT14
003040  003310          .WORD   PAT15
                        ;
                        ;       DATA PATTERNS (EDC OF PATTERN, LENGTH OF PATTERN IN WORDS, FOLLOWED BY PATTERN)
                        ;
003041  177777  PAT0:    .WORD   177777          ; DUMMY EDC
003042  000001          .WORD   1               ; USER DEFINEABLE PATTERN
003043  000000          .WORD   0
003044  000000          .WORD   0
003045  000000          .WORD   0
003046  000000          .WORD   0
003047  000000          .WORD   0
003050  000000          .WORD   0
003051  000000          .WORD   0
003052  000000          .WORD   0
003053  000000          .WORD   0
003054  000000          .WORD   0
003055  000000          .WORD   0
003056  000000          .WORD   0
003057  000000          .WORD   0
003060  000000          .WORD   0
003061  000000          .WORD   0
003062  000000          .WORD   0
003063  115337  PAT1:    .WORD   115337          ; EDC FOR PATTERN 1
003064  000001          .WORD   1               ; B'1000101110001011'
003065  105613          .WORD   105613
003066  010524  PAT2:    .WORD   010524          ; EDC FOR PATTERN 2
003067  000001          .WORD   1
003070  031463          .WORD   031463          ; B'0011001100110011'
003071  001747  PAT3:    .WORD   001747          ; EDC FOR PATTERN 3
003072  000001          .WORD   1
003073  030221          .WORD   030221
003074  135776  PAT4:    .WORD   135776          ; EDC FOR PATTERN 4
003075  000020          .WORD   16.             ; SHIFTING ONES
003076  000001          .WORD   000001
003077  000003          .WORD   000003
003100  000007          .WORD   000007
003101  000017          .WORD   000017
003102  000037          .WORD   000037
003103  000077          .WORD   000077
003104  000177          .WORD   000177
003105  000377          .WORD   000377
```

DATA PATTERNS THAT WILL BE WRITTEN TO THE DRIVE

```
     58 003106  000777                          .WORD   000777
     59 003107  001777                          .WORD   001777
     60 003110  003777                          .WORD   003777
     61 003111  007777                          .WORD   007777
     62 003112  017777                          .WORD   017777
     63 003113  037777                          .WORD   037777
     64 003114  077777                          .WORD   077777
     65 003115  177777                          .WORD   177777
     66 003116  052420                PAT5:     .WORD   052420          ; EDC FOR PATTERN 5
     67 003117  000020                          .WORD   16.             ; SHIFTING ZEROS
     68 003120  177776                          .WORD   177776
     69 003121  177774                          .WORD   177774
     70 003122  177770                          .WORD   177770
     71 003123  177760                          .WORD   177760
     72 003124  177740                          .WORD   177740
     73 003125  177700                          .WORD   177700
     74 003126  177600                          .WORD   177600
     75 003127  177400                          .WORD   177400
     76 003130  177000                          .WORD   177000
     77 003131  176000                          .WORD   176000
     78 003132  174000                          .WORD   174000
     79 003133  170000                          .WORD   170000
     80 003134  160000                          .WORD   160000
     81 003135  140000                          .WORD   140000
     82 003136  100000                          .WORD   100000
     83 003137  000000                          .WORD   000000
     84 003140  114734                PAT6:     .WORD   114734          ; EDC FOR PATTERN 6
     85 003141  000020                          .WORD   16.             ; ALTERNATING ZERO WORD AND ONE WORD IN
     86 003142  000000                          .WORD   000000          ; 3-2-1-1-1 SEQUENCE
     87 003143  000000                          .WORD   000000
     88 003144  000000                          .WORD   000000
     89 003145  177777                          .WORD   177777
     90 003146  177777                          .WORD   177777
     91 003147  177777                          .WORD   177777
     92 003150  000000                          .WORD   000000
     93 003151  000000                          .WORD   000000
     94 003152  177777                          .WORD   177777
     95 003153  177777                          .WORD   177777
     96 003154  000000                          .WORD   000000
     97 003155  177777                          .WORD   177777
     98 003156  000000                          .WORD   000000
     99 003157  177777                          .WORD   177777
    100 003160  000000                          .WORD   000000
    101 003161  177777                          .WORD   177777
    102 003162  140753                PAT7:     .WORD   140753          ; EDC FOR PATTERN 7
    103 003163  000001                          .WORD   1               ; B'1011011011011001'
    104 003164  133331                          .WORD   133331
    105 003165  021147                PAT8:     .WORD   021147          ; EDC FOR PATTERN 8
    106 003166  000020                          .WORD   16.             ; B'0101010101010101' AND
    107 003167  052525                          .WORD   052525          ; B'1010101010101010'
    108 003170  052525                          .WORD   052525          ; IN 3-2-1-1-1 SEQURNCE
    109 003171  052525                          .WORD   052525
    110 003172  125252                          .WORD   125252
    111 003173  125252                          .WORD   125252
    112 003174  125252                          .WORD   125252
    113 003175  052525                          .WORD   052525
    114 003176  052525                          .WORD   052525
```

```
    115 003177  125252                        .WORD   125252
    116 003200  125252                        .WORD   125252
    117 003201  052525                        .WORD   052525
    118 003202  125252                        .WORD   125252
    119 003203  052525                        .WORD   052525
    120 003204  125252                        .WORD   125252
    121 003205  052525                        .WORD   052525
    122 003206  125252                        .WORD   125252
    123 003207  041260              PAT9:     .WORD   041260          ; EDC FOR PATTERN 9
    124 003210  000001                        .WORD   1               ; B'1101101101101100'
    125 003211  155554                        .WORD   155554
    126 003212  074075              PAT10:    .WORD   074075          ; EDC FOR PATTERN 10
    127 003213  000020                        .WORD   16.             ; B'0010110100101101' AND
    128 003214  026455                        .WORD   026455          ; B'1101001011010010'
    129 003215  026455                        .WORD   026455          ; IN 3-2-1-1-1 SEQUENCE
    130 003216  026455                        .WORD   026455
    131 003217  151322                        .WORD   151322
    132 003220  151322                        .WORD   151322
    133 003221  151322                        .WORD   151322
    134 003222  026455                        .WORD   026455
    135 003223  026455                        .WORD   026455
    136 003224  151322                        .WORD   151322
    137 003225  151322                        .WORD   151322
    138 003226  026455                        .WORD   026455
    139 003227  151322                        .WORD   151322
    140 003230  026455                        .WORD   026455
    141 003231  151322                        .WORD   151322
    142 003232  026455                        .WORD   026455
    143 003233  151322                        .WORD   151322
    144 003234  153110              PAT11:    .WORD   153110          ; EDC FOR PATTERN 11
    145 003235  000001                        .WORD   1               ; B'0110110110110110'
    146 003236  066666                        .WORD   066666
    147 003237  046211              PAT12:    .WORD   046211          ; EDC FOR PATTERN 12
    148 003240  000020                        .WORD   16.             ; RIPLE ONE
    149 003241  000001                        .WORD   000001
    150 003242  000002                        .WORD   000002
    151 003243  000004                        .WORD   000004
    152 003244  000010                        .WORD   000010
    153 003245  000020                        .WORD   000020
    154 003246  000040                        .WORD   000040
    155 003247  000100                        .WORD   000100
    156 003250  000200                        .WORD   000200
    157 003251  000400                        .WORD   000400
    158 003252  001000                        .WORD   001000
    159 003253  002000                        .WORD   002000
    160 003254  004000                        .WORD   004000
    161 003255  010000                        .WORD   010000
    162 003256  020000                        .WORD   020000
    163 003257  040000                        .WORD   040000
    164 003260  100000                        .WORD   100000
    165 003261  121147              PAT13:    .WORD   121147          ; EDC FOR PATTERN 13
    166 003262  000020                        .WORD   16.             ; RIPLE ZERO
    167 003263  177776                        .WORD   177776
    168 003264  177775                        .WORD   177775
    169 003265  177773                        .WORD   177773
    170 003266  177767                        .WORD   177767
    171 003267  177757                        .WORD   177757
```

DATA PATTERNS THAT WILL BE WRITTEN TO THE DRIVE

```
    172 003270  177737                        .WORD   177737
    173 003271  177677                        .WORD   177677
    174 003272  177577                        .WORD   177577
    175 003273  177377                        .WORD   177377
    176 003274  176777                        .WORD   176777
    177 003275  175777                        .WORD   175777
    178 003276  173777                        .WORD   173777
    179 003277  167777                        .WORD   167777
    180 003300  157777                        .WORD   157777
    181 003301  137777                        .WORD   137777
    182 003302  077777                        .WORD   077777
    183 003303  125677              PAT14:    .WORD   125677          ; EDC FOR PATTERN 14
    184 003304  000003                        .WORD   3               ; B'1101101101101101'
    185 003305  155555                        .WORD   155555          ; B'1011011011011011' PATTERNS 14 AND 15
    186 003306  133333                        .WORD   133333          ; B'1101101101101101' MAKE TEST 4 FORMATTER
    187 003307  155555                        .WORD   155555          ; REPEATED            COMPATIBLE
    188 003310  030206              PAT15:    .WORD   030206          ; EDC FOR PATTERN 15
    189 003311  000003                        .WORD   3               ; B'1101101101101101'
    190 003312  155555                        .WORD   155555          ; B'1011011011011011'
    191 003313  133333                        .WORD   133333          ; B'0110110110110110'
    192 003314  066666                        .WORD   066666          ; REPEATED
```

```
                                        .SBTTL  ROOT    - MAIN DRIVING MODULE OF TEST 4

003315  104205  037705        ROOT:   MOV     #FIRSTU,R5              ; R5 POINTS TO FIRST UNIT TO BE TESTED
003317  114007                        CLR     R0                      ; COUNT OF DROPPED UNITS IS SET TO ZERO
003320  104155                ROOTLP: MOV     (R5),R5                 ; R5 POINTS TO NEXT UNIT IN CHAIN
003321                                ASSUME  U.NEXT,0
003321  104651  000046                MOV     U.PARM(R5),R1           ; GET UNIT PARAMETERS
003323                                BMI     NOUNIT                  ; IF UNIT DROPPED, BRANCH
        003323  070000  003352        ^O070000,NOUNIT
003325                                ASSUME  DROP,100000             ; ASSUME DROP IS SIGN BIT
003325  115000  003006                TST     M.PARM                  ; SEE IF IN PROCESS OF INITIAL WRITE
003327                                BPL     1$                      ; IF NOT, BRANCH
        003327  030000  003335        ^O030000,1$
003331                                ASSUME  IWIPRG,100000           ; ASSUME IWIPRG IS SIGN BIT
003331  102201  040000                BIT     #INITW,R1               ; SEE IF THIS UNIT IS BEING INITALLY WRITTEN
003333                                BEQ     NOUNIT                  ; IF NOT, BRANCH
        003333  010000  003352        ^O010000,NOUNIT
003335                        1$:     CALL    SEQNCR                  ; CALL SEQUENCER
        003335  020000  003375        ^O020000,SEQNCR
003337  104657  000046                MOV     U.PARM(R5),R0           ; SEE IF JUST DROPPED
003341                                BPL     2$                      ; IF NOT, BRANCH
        003341  030000  003347        ^O030000,2$
003343                                ASSUME  DROP,100000
003343  104203  002410                MOV     #CR.DIS,R3              ; POINT TO DISCONNECT
003345                                CALL    TALK                    ; SEND
        003345  020000  001670        ^O020000,TALK
003347  114007                2$:     CLR     R0                      ; COUNT OF DROPPED UNITS IS SET TO ZERO
003350                                BR      ROOTLP                  ; BRANCH TO BOTTOM OF ROOT
        003350  000000  003320        ^O00,ROOTLP
003352  115407                NOUNIT: INC     R0                      ; INCREMENT COUNT OF DROPPED UNITS
003353  106207  000004                CMP     #4,R0                   ; SEE IF FOUR (ALL) UNITS HAVE BEEN DROPPED
003355                                BNE     ROOTLP                  ; IF NOT, BRANCH
        003355  050000  003320        ^O050000,ROOTLP
003357  115000  003006                TST     M.PARM                  ; SEE IF INITIAL WRITE WAS IN PROGRESS
003361                                BPL     STOP                    ; IF NOT, END TEST4
        003361  030000  003370        ^O030000,STOP
003363                                ASSUME  IWIPRG,100000           ; ASSUME IWIPRG IS SIGN BIT
003363  103200  100000  003006        BIC     #IWIPRG,M.PARM          ; CLEAR INITIAL WRITE BIT
003366                                BR      ROOT                    ; NOW START TESTING
        003366  000000  003315        ^O00,ROOT
003370  104207  060016        STOP:   MOV     #DONE,R0                ; MOVE DONE TO HOST COMMUNICATION AREA
003372                                CALL    HOSTRQ                  ; SENT DONE MESSAGE TO HOST
        003372  020000  001543        ^O020000,HOSTRQ
003374  060021                        XFC     EXIT                    ; EXIT DIAGNOSTIC MACHINE MODE
```

```
                                .SBTTL  SEQNCR - OVERLAY DRIVER FOR TEST 4
003375                          SEQNCR:
                                ;
                                ;       SEQUENCER CALLS THE VARIOUS MODULES, USING THE ADDRESS RETURNED
                                ;       IN R0 BY THE PREVIOUS MODULE.  R0 HOLDS THE NEXT ADDRESS TO VECTOR
                                ;       TO. IF R1 IS ZERO, AN IMMIDATE CALL IS MADE TO THE NEXT ROUTINE.
                                ;       IF R2 IS NONZERO, AN ERROR WAS ENCOUNTERED, AND THE ERROR COUNT IS
                                ;       INCREMENTED.  THE ERROR IS THEN REPORTED TO THE HOST.
                                ;
003375  104657  000013                MOV     U.NFUN(R5),R0   ; LOAD R0 WITH NEXT FUNCTION ADDRESS
003377                          SEQLP:
```

```
                                        .SBTTL  DRPTST - SEE THAT AT LEAST ONE SUBUNIT IS ACTIVE ON A UNIT
                                        ;DRPTST
                                        ;
                                        ;       DROP TEST MAKES SURE AT LEAST ONE ACTIVE SUBUNIT IS ACTIVE ON THE UNIT
                                        ;       IF SO, R4 POINTS TO SUBUNIT PARAMETER BLOCK, U.SUBU IS UPDATED TO
                                        ;       THE SUBUNIT OFFSET, AND R3 IS RETURNED POSITIVE.  IF NOT, R3 IS
                                        ;       RETURNED NEGATIVE AND THE ENTIRE UNIT IS DROPPED
                                        ;
 003377                                         PUSH    R0              ; SAVE NEXT MODULE ADDRESS
 003377  100467                                                                 MOV R0,-(SP)
 003400  104657  000046                         MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
 003402  102207  001000                         BIT     #FTIME,R0       ; SEE IF FIRST TIME THROUGH
 003404                                         BEQ     1$              ; IF NOT, BRANCH
 003404  010000  003504                         ^010000,1$
 003406                                         CALL    GOSEEK          ; IF FIRST TIME, FORCE A SEEK
 003406  020000  005066                         ^020000,GOSEEK
 003410  103207  015610                         BIC     #RBNBN!REVEC!FTIME!DIREC!WCHECK!NEWSUB,R0
 003412  100657  000046                         MOV     R0,U.PARM(R5)   ; SAVE
 003414  104057                                 MOV     R5,R0           ; R0 WILL POINT AT SUBUNIT POINTERS
 003415  115407                                 INC     R0              ; R0 POINTS AT SUBUNIT POINTERS
 003416                                         ASSUME  U.SUBP,1
 003416  114002                                 CLR     R2              ; UP TO 4 SUBUNITS ON THIS UNIT
 003417  100652  000024                         MOV     R2,U.CSEC(R5)   ; ZERO ALL TRACK COUNTS SO THE TEST AT THE
 003421  100652  000021                         MOV     R2,U.NSEC(R5)   ; BEGINNING OF SETUP IS SATISIFIED SO CONTROL
 003423  100652  000023                         MOV     R2,U.TSEC(R5)   ; DROPS THROUGH TO SETUP THE NEXT OPERATION
 003425  104273                         11$:    MOV     (R0)+,R3        ; GET SUBUNIT POINTER
 003426                                         BMI     12$             ; IF NO SUBUNIT, BRANCH
 003426  070000  003443                         ^070000,12$
 003430  104133                                 MOV     (R3),R3         ; GET SUBUNIT PARAMETERS
 003431                                         ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
 003431                                         BMI     12$             ; IF DROPPED, BRANCH
 003431  070000  003443                         ^070000,12$
 003433                                         ASSUME  DROP,100000     ; ASSUME DROP IS SIGN BIT
 003433  115000  003006                         TST     M.PARM          ; SEE IF INTIIAL WRITE IN PROGRESS
 003435                                         BPL     13$             ; IF NOT, BRANCH
 003435  030000  003452                         ^030000,13$
 003437                                         ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
 003437  102203  040000                         BIT     #INITW,R3       ; SEE IF THIS SUBUNIT IS TO BE INITIALLY WRITTEN
 003441                                         BNE     13$             ; IF SO, BRANCH
 003441  050000  003452                         ^050000,13$
 003443  115402                         12$:    INC     R2              ; INCREMENT COUNT
 003444  106202  000003                         CMP     #3,R2           ; SEE IF COUNT EXHAUSTED
 003446                                         BPL     11$             ; IF NOT, BRANCH
 003446  030000  003425                         ^030000,11$
 003450                                         BR      4$              ; IF SO, DROP ENTIRE UNIT
 003450  000000  003700                         ^00,4$
 003452  100652  000050                 13$:    MOV     R2,U.SUBU(R5)   ; SAVE SUBUNIT OFFSET
 003454  104024                                 MOV     R2,R4           ; MOVE TO SUBUNIT POINTER REGISTER
 003455  105054                                 ADD     R5,R4           ; ADD UNIT POINTER
 003456  105204  000001                         ADD     #U.SUBP,R4      ; ADD SUBUNIT POINTER OFFSET
 003460  104144                                 MOV     (R4),R4         ; R4 NOW POINTS TO SUBUNIT
 003461                                         CALL    RECOVR          ; SET UP UNIT FOR RUNNING
 003461  020000  004505                         ^020000,RECOVR
 003463  104207  104000                         MOV     #ERMASK!SEKREQ,R0 ; DISABLE RECOVERY, FORCE A SEEK
 003465  100657  000047                         MOV     R0,U.RCOV(R5)   ; SAVE IN RECOVERY WORD
 003467  104207  000010                         MOV     #NUML2S,R0      ; GET NUMBER OF LEVEL 2 COMMANDS
 003471  104051                                 MOV     R5,R1           ; R1 POINTS TO UNIT PARAMETERS
```

```
     48 003472  105201  000035                    ADD     #U.SDI2,R1      ; R1 POINTS TO SDI LEVEL 2 ERROR COUNT AREA
     49 003474  114002                            CLR     R2              ; SET UP TO CLEAR AREA
     50 003475  100212                 14$:       MOV     R2,(R1)+        ; CLEAR
     51 003476  117407                            DEC     R0              ; DECREMENT COUNT
     52 003477                                    BNE     14$             ; IF INCOMPLETE, BRANCH
        003477  050000  003475                    ^050000,14$
     53 003501  114003                            CLR     R3              ; PROCESS THIS UNIT
     54 003502                                    BR      5$              ; EXIT
        003502  000000  003706                    ^00,5$
     55 003504  104054                 1$:        MOV     R5,R4           ; GET POINTER TO UNIT DATASTRUCTURE
     56 003505  115404                            INC     R4              ; POINT AT SUBUNIT POINTERS
     57 003506                                    ASSUME  U.SUBP,1
     58 003506  105654  000050                    ADD     U.SUBU(R5),R4   ; R4 POINTS AT SPECIFIC SUBUNIT POINTER
     59 003510  104144                            MOV     (R4),R4         ; R4 POINTS AT SUBUNIT PARAMETERS
     60 003511                                    BMI     25$             ; NO SUBUNIT (SOMETIMES HAPPENS IN INITIALIZATION)
        003511  070000  003577                    ^070000,25$
     61 003513  104141                            MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
     62 003514                                    ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
     63 003514  102207  004000                    BIT     #NEWSUB,R0      ; SEE IF SUPPOSED TO GO TO NEXT SUBUNIT
     64 003516                                    BEQ     6$              ; IF NOT, BRANCH
        003516  010000  003567                    ^010000,6$
     65 003520  103207  004000                    BIC     #NEWSUB,R0      ; CLEAR NEWSUB BIT
     66 003522  100657  000046                    MOV     R0,U.PARM(R5)   ; SAVE
     67 003524  114002                            CLR     R2              ; FOR CEARING FOLLOWING WORDS
     68 003525  100652  000024                    MOV     R2,U.CSEC(R5)   ; ZERO ALL TRACK COUNTS SO THE TEST AT THE
     69 003527  100652  000021                    MOV     R2,U.NSEC(R5)   ; BEGINNING OF SETUP IS SATISIFIED SO CONTROL
     70 003531  100652  000023                    MOV     R2,U.TSEC(R5)   ; DROPS THROUGH TO SETUP THE NEXT OPERATION
     71 003533  115000  003006                    TST     M.PARM          ; SEE IF INITIAL WRITE IS IN PROGRESS
     72 003535                                    BPL     2$              ; IF NOT, BRANCH
        003535  030000  003573                    ^030000,2$
     73 003537                                    ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
     74 003537  103201  040000                    BIC     #INITW,R1       ; CLEAR INITIAL WRITE BIT
     75 003541  100141                            MOV     R1,(R4)         ; SAVE
     76 003542                                    ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
     77 003542                                    MSSG    1               ; REPORT INITIAL WRITE COMPLETE
        003542  104200  005527  001577                                            MOV     #MS1,HRQ.02
        003545  100467                                                                    MOV R0,-(SP)
        003546  104650  000063  001576                                            MOV     U.UNUM(R5),HRQ.01
        003551  105650  000050  001576                                            ADD     U.SUBU(R5),HRQ.01
        003554  104207  060C15                                                    MOV     #MESSAG,R0
        003556  020000  001543                    ^020000,HOSTRQ
        003560  104267                                                                    MOV (SP)+,R0
     78 003561  104652  000050                    MOV     U.SUBU(R5),R2   ; GET ACTIVE SUBUNIT POINTER
     79 003563                                    CALL    CHKUP           ; SEE IF ANY MORE UNITS TO INITIAL WRITE
        003563  020000  004756                    ^020000,CHKUP
     80 003565                                    BR      5$              ; BRANCH
        003565  000000  003706                    ^00,5$
     81 003567  11 003                  6$:       CLR     R3              ; IF ACTIVE, FLAG TO PROCESS
     82 003570  115001                            TST     R1              ; SEE IF UNIT ACTIVE
     83 003571                                    BPL     7$              ; IF ACTIVE, BRANCH
        003571  030000  003714                    ^030000,7$
     84 003573                                    ASSUME  DROP,100000     ; ASSUME DROP IS SIGN BIT
     85 003573  102201  040100          2$:       BIT     #SEQSEK!INITW,R1        ; SEE IF SEQUENTIAL SEEKS
     86 003575                                    BNE     10$             ; IF SO, BRANCH
        003575  050000  003644                    ^050000,10$
     87                                 ;RNDSUB
     88                                 ;
```

```
                                        ;       GET NEXT SUBUNIT TO CHECK RANDOMLY
                                        ;
003577  104207  000004          25$:  MOV     #4,R0             ; SUBUNIT COUNT
003601  104203  000001                MOV     #U.SUBP,R3        ; R3 WILL POINT TO SUBUNIT POINTERS
003603  105053                        ADD     R5,R3             ; R3 POINTS TO SUBUNIT POINTERS
003604  104652  000050                MOV     U.SUBU(R5),R2     ; R2 IS SUBUNIT NUMBER
003606  115402                  21$:  INC     R2                ; GO TO NEXT UNIT
003607  106202  000003                CMP     #3,R2             ; SEE IF GREATER THAN 3
003611                                BPL     22$               ; IF NOT, BRANCH
        003611  030000  003614        ^O030000,22$
003613  114002                        CLR     R2                ; NOW ON SUBUNIT 0
003614  104034                  22$:  MOV     R3,R4             ; R4 POINTS TO SUBUNIT POINTERS
003615  105024                        ADD     R2,R4             ; R4 NOW POINTS TO A SPECIFIC SUBUNIT POINTER
003616  104144                        MOV     (R4),R4           ; R4 NOW POINTS TO SUBUNIT PARAMETERS
003617                                BMI     23$               ; IF NO SUBUNIT, BRANCH
        003617  070000  003624        ^O070000,23$
003621  104141                        MOV     (R4),R1           ; GET SUBUNIT PARAMETERS
003622                                ASSUME  S.PARM,0          ; ASSUME THAT S.PARM IS ZERO
003622                                BPL     24$               ; IF ACTIVE, BRANCH
        003622  030000  003637        ^O030000,24$
003624  117407                  23$:  DEC     R0                ; DECREMENT COUNT
003625                                BNE     21$               ; IF INCOMPLETE, BRANCH
        003625  050000  003606        ^O050000,21$
003627  104653  000046                MOV     U.PARM(R5),R3     ; GET UNIT PARAMETERS
003631  101203  100000                BIS     #DROP,R3          ; SET DROP BIT
003633  100653  000046                MOV     R3,U.PARM(R5)     ; SAVE
003635                                BR      5$                ; EXIT
        003635  000000  003706        ^O00,5$
003637  100652  000050          24$:  MOV     R2,U.SUBU(R5)     ; SAVE SUBUNIT NUMBER
003641  114003                        CLR     R3                ; TEST THIS SUBUNIT IMMIDATELY
003642                                BR      5$                ; EXIT
        003642  000000  003706        ^O00,5$
003644  104652  000050          10$:  MOV     U.SUBU(R5),R2     ; R2 IS SUBUNIT NUMBER (0-3)
003646  102207  010000                BIT     #DIREC,R0         ; SEE IF GOING UP OR DOWN
003650                                BNE     3$                ; IF DOWN, BRANCH
        003650  050000  003666        ^O050000,3$
003652                                CALL    CHKUP             ; FIND NEXT UPPER SUBUNIT
        003652  020000  004756        ^O020000,CHKUP
003654  115007                        TST     R0                ; FIND ONE?
003655                                BEQ     5$                ; IF SO, BRANCH
        003655  010000  003706        ^O010000,5$
003657                                CALL    CHKDN             ; SEARCH DOWN FOR ONE
        003657  020000  004721        ^O020000,CHKDN
003661  115007                        TST     R0                ; FIND ONE?
003662                                BEQ     5$                ; IF SO, BRANCH
        003662  010000  003706        ^O010000,5$
003664                                BR      4$                ; DROP UNIT
        003664  000000  003700        ^O00,4$
003666                          3$:   CALL    CHKDN             ; LOOK DOWN FOR SUBUNIT
        003666  020000  004721        ^O020000,CHKDN
003670  115007                        TST     R0                ; FIND ONE?
003671                                BEQ     5$                ; IF SO, BRANCH
        003671  010000  003706        ^O010000,5$
003673                                CALL    CHKUP             ; LOOK UP FOR ONE
        003673  020000  004756        ^O020000,CHKUP
003675  115007                        TST     R0                ; FIND ONE?
003676                                BEQ     5$                ; IF SO, BRANCH
```

```
     003676  010000  003706                    ^010000,5$
131  003700  104653  000046          4$:       MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
132  003702  101203  100000                    BIS     #DROP,R3        ; DROP UNIT
133  003704  100653  000046                    MOV     R3,U.PARM(R5)   ; SAVE
134  003706  115003                  5$:       TST     R3              ; SEE IF THIS SUBUNIT TO BE TESTED
135  003707                                    BNE     7$              ; IF NOT, BRANCH
     003707  050000  003714                    ^050000,7$
136  003711  104200  177777  003002            MOV     #-1,SCR1        ; FLAG AS NEW SUBUNIT
137  003714                  7$:               POP     R0              ; RESTORE R0
     003714  104267                                                    MOV (SP)+,R0
138  003715  115003                            TST     R3              ; SEE IF ACTIVE SUBUNIT EXISTS
139  003716                                    BNE     NOSUB           ; IF SO, BRANCH
     003716  050000  004501                    ^050000,NOSUB
140  003720  104070  003005                    MOV     R0,LSTOVL       ; SAVE THIS OVERLAY NUMBER
```

```
     43 003722  104652  000047          GO4IT:  MOV     U.RCOV(R5),R2   ; GET ERROR RECOVERY WORD
     44 003724                                  BMI     2$              ; IF NO RECOVERY, BRANCH
        003724  070000  003743                  ^O70000,2$
     45 003726                                  ASSUME  ERMASK,100000   ; ASSUME ERMASK SIGN BIT
     46 003726                                  CALL    RTDS            ; BEFORE OPERATION, SEE IF ONLINE
        003726  020000  001417                  ^O20000,RTDS
     47 003730  115002                          TST     R2              ; SEE IF ERROR OCCURRED
     48 003731                                  BNE     1$              ; IF SO, BRANCH
        003731  050000  003737                  ^O50000,1$
     49 003733  102201  000102                  BIT     #AVAIL!ATTN,R1  ; SEE IF AVAILABLE OR ATTENTION ASSERTED
     50 003735                                  BEQ     2$              ; IF NOT, BRANCH
        003735  010000  003743                  ^O10000,2$
     51 003737                          1$:     CALL    GORTRY          ; RETRY THIS MODULE
        003737  020000  005056                  ^O20000,GORTRY
     52 003741                                  BR      JMPRET          ; RECOVER
        003741  000000  003746                  ^O0,JMPRET
     53 003743                          2$:
```

```
      1                                 .SBTTL  JMPRET - THIS IS THE LOCATION THAT VECTORS TO/FROM ALL OVERLAYS
      2 003743                                  PUSH    R0              ; PUSH THE BRANCH ADDRESS ONTO THE STACK
        003743  100467                                                                MOV R0,-(SP)
      3 003744                                  RETURN                  ; VECTOR TO MODULE
        003744  000000  000000                  ^O0,0
      4 003746                          JMPRET:
```

```
                                     .SBTTL  CORECT - CORRECT THE ERRORS
                                     ;CORECT
                                     ;
                                     ;       CORRECT THE ERRORS AND RECOVER
                                     ;
  003746                                     PUSH    <R1,R0>          ; SAVE NEXT OVERLAY AND CALL TYPE
  003746  100461                                                                                MOV R1,-(SP)
  003747  100467                                                                                MOV R0,-(SP)
  003750  114000  003011                     CLR     ERMODE           ; CLEAR ERROR MODES
  003752  115002                             TST     R2               ; SEE IF ERROR OR MBYTE REPORT
  003753                                     BNE     5$               ; IF SO, BRANCH
  003753  050000  004060                     ^050000,5$
  003755  104652  000047                     MOV     U.RCOV(R5),R2    ; GET ERROR RECOVERY WORD
  003757                                     BMI     90$              ; IF RECOVERY DISABLED, BRANCH
  003757  070000  004474                     ^070000,90$
  003761                                     ASSUME  ERMASK,100000    ; ASSUME RECOVERY MASK IS SIGN BIT
  003761                                     CALL    RTDS             ; GET REAL TIME DRIVE STATE
  003761  020000  001417                     ^020000,RTDS
  003763  115002                             TST     R2               ; SEE IF ERROR OCCURPED
  003764                                     BNE     70$              ; IF SO, REPORT
  003764  050000  004115                     ^050000,70$
  003766  102201  000100                     BIT     #AVAIL,R1        ; SEE IF AVAILABLE ASSERTED
  003770                                     BEQ     2$               ; IF NOT, BRANCH
  003770  010000  004024                     ^010000,2$
  003772  102201  000002                     BIT     #ATTN,R1         ; SEE IF SPINABLE
  003774                                     BNE     1$               ; IF SO, BRANCH
  003774  050000  004017                     ^050000,1$
  003776                                     DEVFTL  13               ; REPORT UNSPINABLE DRIVE
  003776  104200  001406  001601                                          MOV     #ER13,HRQ.04
  004001  104202  047655                                                  MOV     #13!FTLDEV+4000.,R2
  004003  104020  001577                                                  MOV     R2,HRQ.02
  004005  104200  004005  001576                                          MOV     #.,HRQ.01
  004010  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
  004013                                     ENDERR  0
  004013  114000  003013                                                  CLR     ERRPOS           ; CLEAR THE POSITION
  004015                                     BR      70$              ; REPORT
  004015  000000  004115                     ^00,70$
  004017  101200  000100  003011     1$:     BIS     #AVAIL,ERMODE    ; AVAIL ERROR
  004022                                     BR      3$               ; BRANCH
  004022  000000  004030                     ^00,3$
  004024  102201  000002             2$:     BIT     #ATTN,R1         ; SEE IF ATTENTION IS ASSERTED
  004026                                     BEQ     90$              ; IF NOT, BRANCH
  004026  010000  004474                     ^010000,90$
  004030  101200  000002  003011     3$:     BIS     #ATTN,ERMODE     ; FLAG AS ATTENTION ERROR
  004033                                     SOFTER  2                ; ATTENTION ASSERTED UNEXPECTEDLY
  004033  104200  000117  001601                                          MOV     #ER2,HRQ.04
  004036  104202  147642                                                  MOV     #2!ERSOFT+4000.,R2
  004040  104020  001577                                                  MOV     R2,HRQ.02
  004042  104200  004042  001576                                          MOV     #.,HRQ.01
  004045  104200  060013  001575                                          MOV     #ERRMES,HRQ.RQ
  004050                                     ENDERR
  004050  104200  000051  001602                                          MOV     #SER22,HRQ.05
  004053  104200  000006  003013                                          MOV     #5+1,ERRPOS      ; SET THE POSITION
  004056                                     BR      6$               ; BRANCH
  004056  000000  004120                     ^00,6$
```

```
      1 004060  106200  060011  001575  5$:     CMP     #T4MXFR,HRQ.RQ  ; SEE IF MEGABIT TRANSFER REPORT
      2 004063                                  BNE     40$             ; IF NOT, BRANCH
        004063  050000  004105                  ^O50000,40$
      3 004065  104650  000063  001576          MOV     U.UNUM(R5),HRQ.01       ; MOVE STARTING UNIT NUMBER TO COM AREA
      4 004070  105650  000050  001576          ADD     U.SUBU(R5),HRQ.01       ; ADD OFFSET
      5 004073  104307  001575                  MOV     HRQ.RQ,R0       ; SETUP FOR COM
      6 004075                                  CALL    HOSTRQ          ; REPORT
        004075  020000  001543                  ^O20000,HOSTRQ
      7 004077  115000  001576                  TST     HRQ.01          ; SEE IF DROPPED
      8 004101                                  BMI     80$             ; IF SO, BRANCH
        004101  070000  004434                  ^O70000,80$
      9 004103                                  BR      90$             ; IF NOT, BRANCH
        004103  000000  004474                  ^O0,90$
```

```
      1 004105  104302  001577          40$:    MOV     HRQ.02,R2       ; GET ERROR TYPE
      2 004107  103202  037777                  BIC     #37777,R2       ; CLEAR ERROR NUMBER
      3 004111  106202  040000                  CMP     #FTLDEV,R2      ; SEE IF DEVICE FATAL
      4 004113                                  BNE     6$              ; IF NOT, BRANCH
        004113  050000  004120                  ^050000,6$
      5 004115                                  ASSUME  FTLDEV,040000   ; ASSUME IN HI 2 BITS
      6 004115  101200  100000  003011  70$:    BIS     #DROP,ERMODE    ; ERROR CAUSED THE ENTIRE DEVICE TO BE DROPPED
      7 004120  104307  003013          6$:     MOV     ERRPOS,R0       ; GET ERROR POSITION
      8 004122                                  BEQ     20$             ; IF NO REPORTING, BRANCH
        004122  010000  004303                  ^010000,20$
      9 004124  103207  100000                  BIC     #VALID,R0       ; CLEAR 'ALLREADY VALID' BIT
     10 004126  105207  001576                  ADD     #HRQ.RQ+1,R0    ; POINT TO OUTPUT BUFFER
     11 004130                                  PUSH    <R4,R0>         ; SAVE SUBUNIT POINTER AND ERROR POINTER
        004130  100464                                                                  MOV R4,-(SP)
        004131  100467                                                                  MOV R0,-(SP)
     12 004132  104657  000011                  MOV     U.MSTO(R5),R0   ; GET MASTER SEEK TIMEOUT
     13 004134  115407                          INC     R0              ; INCREMENT R0 FOR BNE RATHER THAN BCC
     14 004135  114003                          CLR     R3              ; FOR LOW ORDER TIMEOUT
     15 004136                          31$:    CALL    RTDSL           ; GET REAL TIME DRIVE STATE
        004136  020000  001453                  ^020000,RTDSL
     16 004140  115002                          TST     R2              ; SEE IF ERROR
     17 004141                                  BNE     33$             ; IF ERROR, BRANCH
        004141  050000  004162                  ^050000,33$
     18 004143  102201  000003                  BIT     #RCVRDY!ATTN,R1 ; SEE IF RECEIVER READY OR ATTENTION ASSERTED
     19 004145                                  BNE     33$             ; IF SO, BRANCH
        004145  050000  004162                  ^050000,33$
     20 004147  104201  000310                  MOV     #200.,R1        ; INNER LOOP TIMEOUT
     21 004151  117401                  32$:    DEC     R1              ; DECREMENT COUNT
     22 004152                                  BNE     32$             ; IF INCOMPLETE, BRANCH
        004152  050000  004151                  ^050000,32$
     23 004154  117403                          DEC     R3              ; DECREMENT TIMEOUT
     24 004155                                  BNE     31$             ; LOOP IF INCOMPLETE
        004155  050000  004136                  ^050000,31$
     25 004157  117407                          DEC     R0              ; DECREMENT TIMEOUT
     26 004160                                  BNE     31$             ; IF INCOMPLETE, BRANCH
        004160  050000  004136                  ^050000,31$
     27 004162  104657  000047  33$:    MOV     U.RCOV(R5),R0   ; GET RECOVERY TYPE
     28 004164  102207  040000                  BIT     #DRINIT,R0      ; SEE IF DRIVE WAS INITIALIZED
     29 004166                                  BEQ     34$             ; IF NOT, BRANCH
        004166  010000  004176                  ^010000,34$
     30 004170  101207  040000                  BIS     #DRINIT,R0      ; CLEAR INITIALIZE FLAG
     31 004172  100657  000047                  MOV     R0,U.RCOV(R5)   ; SAVE RECOVERY STATUS
     32 004174  104301  002755                  MOV     HIBN,R1         ; GET ORIGINAL STATE
     33 004176  104167                  34$:    MOV     (SP),R0         ; RESTORE ERROR POINTER
     34 004177  100471                          MOV     R1,-(R0)        ; SAVE STATE IN ERROR MESSAGE
     35 004200  115000  003013                  TST     ERRPOS          ; SEE IF STATUS VALID
     36 004202                                  BMI     26$             ; IF SO, BRANCH
        004202  070000  004270                  ^070000,26$
     37 004204                                  ASSUME  VALID,100000    ; ASSUME VALID BIT IS MSB
     38 004204                                  CALL    RTDSL           ; GET REAL TIME DRIVE STATE
        004204  020000  001453                  ^020000,RTDSL
     39 004206  115002                          TST     R2              ; SEE IF VALID
     40 004207                                  BNE     24$             ; IF NOT, BRANCH
        004207  050000  004254                  ^050000,24$
     41 004211  102201  000101                  BIT     #RCVRDY!AVAIL,R1 ; SEE IF RECEIVER READY OR AVAILABLE
     42 004213                                  BEQ     24$             ; IF NOT, BRANCH
        004213  010000  004254                  ^010000,24$
```

```
     43 004215  104204  001750                    MOV     #MAXSND,R4      ; SET UP TIMEOUT
     44 004217  104203  002357                    MOV     #CR.GST,R3      ; POINT TO GET STATUS COMMAND
     45 004221  104652  000025                    MOV     U.MASK(R5),R2   ; GET UNIT SDI SELECT MASK
     46 004223  104137                    21$:    MOV     (R3),R0         ; POINTS TO SDI COMMAND BUFFER
     47 004224                                    ASSUME  L2.OPC,0
     48 004224  104631  000001                    MOV     L2.SLN(R3),R1   ; LOAD BYTE COUNT
     49 004226  060004                            XFC     SEND            ; SEND SDI COMMAND
     50 004227  115001                            TST     R1              ; SEE IF SDI COMMAND SENT SUCESSFULLY
     51 004230                                    BEQ     22$             ; IF SO, BRANCH
        004230  010000  004237                    ^010000,22$
     52 004232  117404                            DEC     R4              ; DECREMENT TIMEOUT
     53 004233                                    BNE     21$             ; IF TIMEOUT INCOMPLETE, BRANCH
        004233  050000  004223                    ^050000,21$
     54 004235                                    BR      24$             ; BRANCH
        004235  000000  004254                    ^00,24$
     55 004237  104654  000033            22$:    MOV     U.SDIS(R5),R4   ; R4 HAS SHORT TIMEOUT
     56 004241  104207  002442            23$:    MOV     #ST,R0          ; POINT TO RECEIVE BUFFER
     57 004243  104631  000002                    MOV     L2.RLN(R3),R1   ; NUMBER OF WORDS IN RESPONSE
     58 004245  060005                            XFC     RCV             ; RECEIVE SDI RESPONSE
     59 004246  115001                            TST     R1              ; SEE IF SDI RESPONSE RECEIVED SUCESSFULLY
     60 004247                                    BEQ     26$             ; IF SO, BRANCH
        004247  010000  004270                    ^010000,26$
     61 004251  117404                            DEC     R4              ; DECREMENT TIMEOUT VALUE
     62 004252                                    BNE     23$             ; IF TIMEOUT UNEXPIRED, BRANCH
        004252  050000  004241                    ^050000,23$
     63 004254                            24$:    POP     <R0,R4>         ; RESTORE ERROR AND SUBUNIT POINTER
        004254  104267                                                                       MOV (SP)+,R0
        004255  104264                                                                       MOV (SP)+,R4
     64 004256  104203  177777                    MOV     #-1,R3          ; R3 CONTAINS 'UNABLE TO GET STATUS'
     65 004260  104202  000007                    MOV     #7,R2           ; R2 CONTAINS COUNT
     66 004262  100273                    25$:    MOV     R3,(R0)+        ; MOVE TO OUTPUT BUFFER
     67 004263  117402                            DEC     R2              ; DECREMENT COUNT
     68 004264                                    BNE     25$             ; IF COUNT INCOMPLETE, BRANCH
        004264  050000  004262                    ^050000,25$
     69 004266                                    BR      8$              ; BRANCH
        004266  000000  004323                    ^00,8$
     70 004270                            26$:    POP     <R0,R4>         ; RESTORE ERROR AND SUBUNIT POINTER
        004270  104267                                                                       MOV (SP)+,R0
        004271  104264                                                                       MOV (SP)+,R4
     71 004272  104201  002451                    MOV     #ST+7,R1        ; R1 POINTS TO AFTER STATUS BUFFER
     72 004274  104202  000007                    MOV     #7,R2           ; R2 HAS COUNT
     73 004276  104413                    27$:    MOV     -(R1),R3        ; MOVE STATUS WORD TO R3
     74 004277  100273                            MOV     R3,(R0)+        ; MOVE TO OUTPUT BUFFER
     75 004300  117402                            DEC     R2              ; DECREMENT COUNT
     76 004301                                    BNE     27$             ; BRANCH IF COUNT INCOMPLETE
        004301  050000  004276                    ^050000,27$
     77 004303  102200  000002  003011  20$:     BIT     #ATTN,ERMODE    ; SEE IF UNEXPECTED ATTENTION
     78 004306                                    BEQ     8$              ; IF NOT, BRANCH
        004306  010000  004323                    ^010000,8$
     79                                   ;       SEE IF ERROR WAS CAUSE OF ATTENTION
     80 004310  104301  002444                    MOV     ST+ST.ERR,R1    ; GET ERROR BYTE
     81 004312  103201  177407                    BIC     #<HIBYTE!7>,R1  ; CLEAR UNUSED BITS
     82 004314                                    BNE     8$              ; IF ERROR(S), BRANCH
        004314  050000  004323                    ^050000,8$
     83                                   ;       SEE IF LOGGABLE INFORMATION IS CAUSE OF ATTENTION
     84 004316  102200  000010  002443            BIT     #ST.EL,ST+ST.REQ        ; SEE IF LOGGABLE INFO
     85 004321                                    BEQ     50$             ; IF NOT, BRANCH ('RECOVER')
```

```
        004321  010000  004375                  ^010000,50$
```

```
      1 004323  104650  000063  001600  8$:     MOV     U.UNUM(R5),HRQ.03       ; GET BASE NUMBER
      2 004326  105650  000050  001600          ADD     U.SUBU(R5),HRQ.03       ; ADD SUBUNIT OFFSET
      3 004331  104307  001575                  MOV     HRQ.RQ,R0       ; MOVE REQUEST TO R0
      4 004333                                  CALL    HOSTRQ          ; REPORT TO HOST
        004333  020000  001543                  ^O020000,HOSTRQ
      5 004335  115000  001576                  TST     HRQ.01          ; SEE IF UNIT DROPPED
      6 004337                                  BMI     80$             ; IF SO, BRANCH
        004337  070000  004434                  ^O070000,80$
      7 004341                                  ASSUME  DROP,100000     ; HOST DROP BIT = BIT 15
      8 004341  104652  000047                  MOV     U.RCOV(R5),R2   ; SEE IF RECOVERY IS ENABLED
      9 004343                                  BMI     90$             ; IF NOT, BRANCH
        004343  070000  004474                  ^O070000,90$
     10 004345                                  ASSUME  ERMASK,100000   ; ASSUME RECOVERY MASK IS SIGN BIT
     11 004345  102200  000002  003011          BIT     #ATTN,ERMODE    ; DID WE HAVE AN ERROR?
     12 004350                                  BEQ     50$             ; IF NOT, BRANCH
        004350  010000  004375                  ^O010000,50$
     13 004352                                  REPSFT  SOFT            ; ELSE, REPORT SOFT ERROR
        004352  104200  000001  001577                                          MOV     #1,HRQ.02
        004355  114000  001600                                                  CLR     HRQ.03
        004357  114000  001601                                                  CLR     HRQ.04
        004361  100467                                                                  MOV R0,-(SP)
        004362  104657  000063                                                  MOV     U.UNUM(R5),R0
        004364  105657  000050                                                  ADD     U.SUBU(R5),R0
        004366  104070  001576                                                  MOV     R0,HRQ.01
        004370  104207  060007                                                  MOV     #T4SOFT,R0
        004372  020000  001543                  ^O020000,HOSTRQ
        004374  104267                                                                  MOV (SP)+,R0
     14 004375                          50$:    CALL    RECOVR          ; RECOVER FROM ERRORS
        004375  020000  004505                  ^O020000,RECOVR
     15 004377  115002                          TST     R2              ; SEE IF ERRORS OCCURRED
     16 004400                                  BNE     40$             ; IF SO, LOOP
        004400  050000  004105                  ^O050000,40$
     17 004402                                  POP     <R0,R1>         ; RESTORE R0, R1
        004402  104267                                                                  MOV (SP)+,R0
        004403  104261                                                                  MOV (SP)+,R1
     18 004404  104653  000047                  MOV     U.RCOV(R5),R3   ; GET RECOVERY WORD
     19 004406  104032                          MOV     R3,R2           ; STORE IN R2
     20 004407  103202  001000                  BIC     #RETRY,R2       ; CLEAR RETRY BIT
     21 004411  100652  000047                  MOV     R2,U.RCOV(R5)   ; STORE RECOVERY WORD
     22 004413  102203  006000                  BIT     #SEKREQ!RCBREQ,R3       ; SEE IF SEEK NEEDED
     23 004415                                  BEQ     9$              ; IF NOT, BRANCH
        004415  010000  004423                  ^O010000,9$
     24 004417  104207  015642                  MOV     #SEEK,R0        ; SEEK NEXT
     25 004421                                  BR      NOSUB           ; EXIT
        004421  000000  004501                  ^O00,NOSUB
     26 004423  102203  001000          9$:     BIT     #RETRY,R3       ; SEE IF LAST MODULE SHOULD BE RETRIED
     27 004425                                  BEQ     100$            ; IF NOT, BRANCH
        004425  010000  004476                  ^O010000,100$
     28 004427  104307  003005                  MOV     LSTOVL,R0       ; RETRY LAST OVERLAY
     29 004431  114001                          CLR     R1              ; IMMIDATELY
     30 004432                                  BR      100$            ; EXIT
        004432  000000  004476                  ^O00,100$
     31 004434                          80$:    POP     <R0,R1>         ; RESTORE REGISTERS
        004434  104267                                                                  MOV (SP)+,R0
        004435  104261                                                                  MOV (SP)+,R1
     32 004436                                  CALL    DSABLE          ; DISABLE ERROR RECOVERY
        004436  020000  005046                  ^O020000,DSABLE
```

```
     33 004440  115000  003011                    TST     ERMODE          ; SEE IF ENTIRE UNIT DROPPED
     34 004442                                    BPL     10$             ; IF NOT, BRANCH
        004442  030000  004456                    ^O030000,10$
     35 004444                                    ASSUME  DROP,100000     ; ASSUME DROP SIGN BIT
     36 004444  114001                            CLR     R1              ; IMMIDATE CALL
     37 004445  104207  016727                    MOV     #DRPALL,R0      ; DRPALL NEXT
     38 004447  102200  000001  003006            BIT     #INTINP,M.PARM  ; SEE IF INITIALIZATION IN PROGRESS
     39 004452                                    BEQ     100$            ; IF NOT, BRANCH
        004452  010000  004476                    ^O010000,100$
     40 004454                                    BR      NOSUB           ; IF SO, JUST RETURN TO TROOT
        004454  000000  004501                    ^O00,NOSUB
     41 004456  104142                  10$:     MOV     (R4),R2         ; GET SUBUNIT PARAMETERS
     42 004457                                    ASSUME  S.PARM,0
     43 004457  101202  100000                    BIS     #DROP,R2        ; DROP SUBUNIT
     44 004461  100142                            MOV     R2,(R4)         ; SAVE
     45 004462                                    ASSUME  S.PARM,0
     46 004462  104651  000046                    MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
     47 004464  101201  001000                    BIS     #FTIME,R1       ; MARK AS FIRST TIME
     48 004466  100651  000046                    MOV     R1,U.PARM(R5)   ; SAVE
     49 004470  104207  005134                    MOV     #SETUP,R0       ; SETUP NEXT MODULE
     50 004472                                    BR      NOSUB           ; EXIT
        004472  000000  004501                    ^O0,NOSUB
     51 004474                          90$:     POP     <R0,R1>         ; RESTORE REGISTERS
        004474  104267                                                                  MOV (SP)+,R0
        004475  104261                                                                  MOV (SP)+,R1
     52 004476  115001                  100$:    TST     R1              ; SEE IF IMMIDATE CALL
     53 004477                                    BEQ     SEQLP           ; IF SO, BRANCH
        004477  010000  003377                    ^O010000,SEQLP
     54 004501  100657  000013          NOSUB:   MOV     R0,U.NFUN(R5)   ; SAVE NEXT FUNCTION
     55 004503                                    RETURN                  ; RETURN TO ROOT
        004503  000000  000000                    ^O00,0
     56
```

```
                                        .SBTTL  RECOVR - RECOVER FROM THE ERROR
004505                                  RECOVR:
                                        ;
                                        ;       SET UP DRIVE AS IT SHOULD BE
                                        ;
004505                                          CALL    RTDS            ; SEE IF WE CAN GET VALID STATE
004505  020000  001417                          ^O020000,RTDS
004507  115002                                  TST     R2              ; SEE IF WE HAVE VALID STATE
004510                                          BNE     10$             ; IF NOT, BRANCH
004510  050000  004715                          ^O050000,10$
004512  104203  002352                          MOV     #CR.ONL,R3      ; POINT TO ONLINE COMMAND
004514                                          CALL    TALK            ; INITIATE SDI INTERCHANGE
004514  020000  001670                          ^O020000,TALK
004516  115002                                  TST     R2              ; SEE IF ERROR OCCURRED
004517                                          BEQ     1$              ; IF NOT, BRANCH
004517  010000  004526                          ^O010000,1$
004521                                          CERROR  5,#SER2         ; REPORT SECONDARY ERROR
004521  104200  004665  001602                                                  MOV     #SER2,HRQ.05
004524                                          BR      10$             ; BRANCH TO ERROR LOOP
004524  000000  004715                          ^O00,10$
004526  104203  002357                  1$:     MOV     #CR.GST,R3      ; POINT TO GET STATUS
004530                                          CALL    TALK            ; SEND SDI INTERCHANGE
004530  020000  001670                          ^O020000,TALK
004532  115002                                  TST     R2              ; SEE IF AN ERROR OCCURRED
004533                                          BEQ     2$              ; BRANCH IF NO ERROR
004533  010000  004542                          ^O010000,2$
004535                                          CERROR  5,#SER0         ; REPORT SECONDARY ERROR
004535  104200  004630  001602                                                  MOV     #SER0,HRQ.05
004540                                          BR      10$             ; LOOP AND TRY AGAIN
004540  000000  004715                          ^O00,10$
004542  104207  177777                  2$:     MOV     #-1,R0          ; SET UP FOR COMPLEMENT OF STATUS
004544  103307  002443                          BIC     ST+ST.REQ,R0    ; R0 HAS STATUS
004546  102207  000023                          BIT     #ST.PS!ST.SR!ST.RU,R0   ; SEE IF ANY FATAL STATES EXIST
004550                                          BEQ     6$              ; IF NOT, BRANCH
004550  010000  004627                          ^O010000,6$
004552                                          DEVFTL  23              ; REPORT SWITCH OUT OR SPINDLE NOT READY
004552  104200  002157  001601                                                  MOV     #ER23,HRQ.04
004555  104202  047667                                                          MOV     #23!FTLDEV+4000.,R2
004557  104020  001577                                                          MOV     R2,HRQ.02
004561  104200  004561  001576                                                  MOV     #.,HRQ.01
004564  104200  060014  001575                                                  MOV     #ERRMC,HRQ.RQ
004567  102207  000001                          BIT     #ST.RU,R0       ; SEE IF RUN/STOP SWITCH OUT
004571                                          BEQ     3$              ; IF NOT, BRANCH
004571  010000  004600                          ^O010000,3$
004573                                          CERROR  5,#SER42        ; REPORT RUN SWITCH OUT
004573  104200  002200  001602                                                  MOV     #SER42,HRQ.05
004576                                          BR      5$              ; BRANCH
004576  000000  004614                          ^O00,5$
004600  102207  000020                  3$:     BIT     #ST.SR,R0       ; SEE IF SPINDLE DROPPED READY
004602                                          BEQ     4$              ; IF NOT, BRANCH
004602  010000  004611                          ^O010000,4$
004604                                          CERROR  5,#SER43        ; REPORT SPINDLE NOT READY
004604  104200  002214  001602                                                  MOV     #SER43,HRQ.05
004607                                          BR      5$              ; BRANCH
004607  000000  004614                          ^O00,5$
004611                                  4$:     CERROR  5,#SER44        ; REPORT PORT SWITCH OUT
004611  104200  002231  001602                                                  MOV     #SER44,HRQ.05
```

```
                                5$:     ENDERR
                                                                        MOV     #SER22,HRQ.06
                                                                        MOV     #6+1,ERRPOS     ; SET THE POSITION
004614 104200 000051 001603
004617 104200 000007 003013
004622 101200 100000 003013             BIS     #100000,ERRPOS  ; STATUS VALID
004625                                  BR      10$             ; BRANCH
004625 000000 004715                    ^O0,10$
004627 102207 000100            6$:     BIT     #ST.RR,R0       ; SEE IF DRIVE IS TO BE RECALIBRATED
004631                                  BNE     7$              ; IF NOT, BRANCH (NOTE COMPLEMENT)
004631 050000 004635                    ^O50000,7$
004633                                  CALL    GORCLB          ; MARK AS RECALIBRATION NEEDED
004633 020000 005076                    ^O20000,GORCLB
004635 104307 002443            7$:     MOV     ST+ST.MOD,R0    ; GET MODE BITS
004637 110707                           SWAB    R0              ; MOVE WRITE PROTECT BUTTON STATUS TO LO BYTE
004640 103207 177417                    BIC     #LBHINB,R0      ; CLEAR UNUSED BITS
004642 100657 000026                    MOV     R0,U.WRIT(R5)   ; SAVE WRITE PROTECT STATUS FOR CHANGE MODE
004644 104301 002444                    MOV     ST+ST.ERR,R1    ; GET ERROR BITS
004646 103201 177400                    BIC     #HIBYTE,R1      ; CLEAR UNUSED BITS
004650                                  BEQ     9$              ; IF NO ERRORS, BRANCH
004650 010000 004676                    ^O10000,9$
004652 102201 000200                    BIT     #ST.DE,R1       ; SEE IF A DRIVE ERROR OCCURRED
004654                                  BEQ     11$             ; IF NOT, BRANCH
004654 010000 004660                    ^O10000,11$
004656                                  CALL    GOSEEK          ; SEEK IS REQUIRED
004656 020000 005066                    ^O20000,GOSEEK
004660 104010 002427            11$:    MOV     R1,DCLR         ; MOVE ERROR BITS TO CLEAR DRIVE COMMAND
004662 104203 002364                    MOV     #CR.CLR,R3      ; POINT TO SDI COMMAND
004664                                  CALL    TALK            ; INITIATE SDI INTERCHANGE
004664 020000 001670                    ^O20000,TALK
004666 115002                           TST     R2              ; SEE IF ANY ERRORS OCCURRED
004667                                  BEQ     9$              ; IF NOT, BRANCH
004667 010000 004676                    ^O10000,9$
004671                                  CERROR  5,#SER1         ; REPORT SECONDARY ERROR
                                                                        MOV     #SER1,HRQ.05
004671 104200 004646 001602
004674                                  BR      10$             ; BRANCH TO ERROR LOOP
004674 000000 004715                    ^O0,10$
004676 104653 000026            9$:     MOV     U.WRIT(R5),R3   ; GET WRITE PROTECTION BUTTON STATUS
004700 101653 000045                    BIS     U.WPRT(R5),R3   ; SET WRITE PROT BITS FOR READ ONLY DRIVES
004702 101203 177402                    BIS     #177402,R3      ; SET OTHER CHANGE MODE BITS
004704 104030 002425                    MOV     R3,WRITBT       ; MOVE TO CHANGE MODE COMMAND
004706 104203 002371                    MOV     #CR.MOD,R3      ; POINT TO MODE COMMAND
004710                                  CALL    TALK            ; INITIATE SDI INTERCHANGE
004710 020000 001670                    ^O20000,TALK
004712                                  CERROR  5,#SER3         ; REPORT SECONDARY ERROR (IF NO ERROR, DON'T CARE)
                                                                        MOV     #SER3,HRQ.05
004712 104200 004707 001602
004715 114000 003011            10$:    CLR     ERMODE          ; RESET ERMODE
004717                                  RETURN                  ; RETURN TO CALLING PROGRAM
004717 000000 000000                    ^O0,0
```

```
                                        .SBTTL  CHKDN  - FIND PREVIOUS ACTIVE SUBUNIT
004721                                  CHKDN:
                                        ;
                                        ;       SEARCH THE SUBUNIT LIST DOWN AND TRY TO FIND AN ACTIVE SUBUNIT
                                        ;
004721  104203  000001                          MOV     #U.SUBP,R3      ; R3 WILLPOINT TO SUBUNIT POINTERS
004723  105053                                  ADD     R5,R3           ; R3 POINTS TO SUBUNIT POINTERS
004724  117402                          1$:     DEC     R2              ; DECREMENT OLD SUBUNIT
004725                                          BMI     2$              ; IF ALL SUBUNITS CHECKED, BRANCH
004725  070000  004745                          ^070000,2$
004727  104034                                  MOV     R3,R4           ; MOVE SUBUNIT LIST POINTER TO R4
004730  105024                                  ADD     R2,R4           ; POINT TO SUBUNIT
004731  104144                                  MOV     (R4),R4         ; R4 POINTS TO SUBUNIT
004732                                          BMI     1$              ; IF NO UNIT, BRANCH
004732  070000  004724                          ^070000,1$
004734  104147                                  MOV     (R4),R0         ; R0 HAS SUBUNIT PARAMETERS
004735                                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
004735                                          BMI     1$              ; IF THIS SUBUNIT DROPPED, BRANCH
004735  070000  004724                          ^070000,1$
004737                                          ASSUME  DROP,100000     ; ASSUME DROP IS SIGN BIT
004737  100652  000050                          MOV     R2,U.SUBU(R5)   ; SAVE SUBUNIT NUMBER
004741  114007                                  CLR     R0              ; FOUND A SUBUNIT
004742  114003                                  CLR     R3              ; PROCESS THIS SUBUNIT
004743                                          BR      3$              ; EXIT
004743  000000  004754                          ^00,3$
004745  104657  000046                  2$:     MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
004747  103207  010000                          BIC     #DIREC,R0       ; SET DIRECTION UP
004751  100657  000046                          MOV     R0,U.PARM(R5)   ; SAVE
004753  104057                                  MOV     R5,R0           ; DIDN'T FIND A SUBUNIT
004754                                  3$:     RETURN                  ; RETURN TO CALLING PROGRAM
004754  000000  000000                          ^00,0
```

```
                                .SBTTL  CHKDN  - FIND FOLLOWING ACTIVE SUBUNIT
004756                          CHKUP:
                                ;
                                ;       SEARCH THE SUBUNIT LIST UP AND TRY TO FIND AN ACTIVE SUBUNIT
                                ;
004756  104203  000001                MOV     #U.SUBP,R3      ; R3 WILLPOINT TO SUBUNIT POINTERS
004760  105053                        ADD     R5,R3           ; R3 POINTS TO SUBUNIT POINTERS
004761  115402                1$:     INC     R2              ; INCREMENT OLD SUBUNIT
004762  106202  000003                CMP     #3,R2           ; SEE IF ALL SUBUNITS CHECKED
004764                                BMI     3$              ; IF ALL SUBUNITS CHECKED, BRANCH
004764  070000  005014                ^O70000,3$
004766  104034                        MOV     R3,R4           ; MOVE SUBUNIT LIST POINTER TO R4
004767  105024                        ADD     R2,R4           ; POINT TO SUBUNIT
004770  104144                        MOV     (R4),R4         ; R4 POINTS TO SUBUNIT
004771                                BMI     1$              ; IF NO UNIT, BRANCH
004771  070000  004761                ^O70000,1$
004773  104147                        MOV     (R4),R0         ; R0 HAS SUBUNIT PARAMETERS
004774                                ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
004774                                BMI     1$              ; IF THIS SUBUNIT DROPPED, BRANC
004774  070000  004761                ^O70000,1$
004776                                ASSUME  DROP,100000     ; ASSUME DROP IS SIGN BIT
004776  115000  003006                TST     M.PARM          ; SEE IF INITIAL WRITE IN PROGRESS
005000                                BPL     2$              ; IF NOT, BRANCH
005000  030000  005006                ^O30000,2$
005002                                ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
005002  102207  040000                BIT     #INITW,R0       ; SEE IF INITIAL WRITE TO BE DONE ON SUBUNIT
005004                                BEQ     1$              ; IF NOT, BRANCH
005004  010000  004761                ^O10000,1$
005006  100652  000050        2$:     MOV     R2,U.SUBU(R5)   ; SAVE SUBUNIT NUMBER
005010  114007                        CLR     R0              ; FOUND ONE
005011  114003                        CLR     R3              ; PROCESS THIS ONE
005012                                BR      5$              ; EXIT
005012  000000  005034                ^O00,5$
005014  104651  000046        3$:     MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
005016  115000  003006                TST     M.PARM          ; SEE IF DOING AN INITIAL WRITE
005020                                BPL     4$              ; IF NOT, BRANCH
005020  030000  005030                ^O30000,4$
005022                                ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
005022  103201  040000                BIC     #INITW,R1       ; FLAG INITIAL WRITE AS COMPLETE ON THIS UNIT
005024  101201  001000                BIS     #FTIME,R1       ; START TESTING UNIT
005026                                BR      9$              ; EXIT
005026  000000  005032                ^O00,9$
005030  101201  010000        4$:     BIS     #DIREC,R1       ; SET DIRECTON DOWN
005032  100651  000046        9$:     MOV     R1,U.PARM(R5)   ; SAVE
005034                          5$:   RETURN                  ; RETURN TO CALLING PROGRAM
005034  000000  000000                ^O00,0
```

```
                                .SBTTL  ERROR RECOVERY SUPPORT SUBROUTINES
                                .SBTTL  ENABLE - ENABLE ERROR RECOVERY
005036                          ENABLE:
                                ;
                                ;       ENABLE ERROR RECOVERY
                                ;
005036  104653  000047                  MOV     U.RCOV(R5),R3   ; GET RECOVERY WORD
005040  103203  100000                  BIC     #ERMASK,R3      ; CLEAR MASK
005042  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
005044                                  RETURN                  ; RETURN
005044  000000  000000                  ^O0,0
                                .SBTTL  DSABLE - DISABLE ERROR RECOVERY
005046                          DSABLE:
                                ;
                                ;       DISABLE ERROR RECOVERY
                                ;
005046  104653  000047                  MOV     U.RCOV(R5),R3   ; GET RECOVERY WORD
005050  101203  100000                  BIS     #ERMASK,R3      ; SET MASK
005052  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
005054                                  RETURN                  ; RETURN
005054  000000  000000                  ^O0,0
                                .SBTTL  GORTRY - RETRY OF MODULE REQUIRED TO RECOVER
005056                          GORTRY:
                                ;
                                ;       JUST RETRY THE LAST MODULE TO RECOVER
                                ;
005056                                  CALL    ENABLE          ; ENABLE ERROR RECOVERY
005056  020000  005036                  ^O020000,ENABLE
005060  101203  001000                  BIS     #RETRY,R3       ; SET RETRY BIT
005062  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
005064                                  RETURN                  ; RETURN
005064  000000  000000                  ^O0,0
                                .SBTTL  GOSEEK - SEEK REQUIRED TO RECOVER
005066                          GOSEEK:
                                ;
                                ;       GOSEEK WILL FLAG THAT THE DRIVE MUST SEEK TO RECOVER FROM AN ERROR
                                ;
005066                                  CALL    ENABLE          ; ENABLE ERROR RECOVERY
005066  020000  005036                  ^O020000,ENABLE
005070  101203  004000                  BIS     #SEKREQ,R3      ; SET SEEK REQUIRED BIT
005072  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
005074                                  RETURN                  ; RETURN
005074  000000  000000                  ^O0,0
                                .SBTTL  GORCLB - RECALIBRATE REQUIRED TO RECOVER
005076                          GORCLB:
                                ;
                                ;       GORCLB WILL FLAG THAT THE DRIVE MUST RECALIBRATE TO RECOVER FROM AN ERROR
                                ;
005076                                  CALL    ENABLE          ; ENABLE ERROR RECOVERY
005076  020000  005036                  ^O020000,ENABLE
005100  101203  006000                  BIS     #SEKREQ!RCBREQ,R3       ; SET SEEK AND RECALIBRATE REQUIRED BIT
005102  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
005104                                  RETURN                  ; RETURN
005104  000000  000000                  ^O0,0
                                .SBTTL  GOINIT - DRIVE MUST BE INITIALIZED TO RECOVER
005106                          GOINIT:
                                ;
```

```
                                ;       INIT THE DRIVE, THEN FLAG AS DONE
                                ;
005106                          ;       PUSH    R1              ; SAVE R1 (DON'T KNOW IF NEEDED)
005106  100461                                                                 MOV R1,-(SP)
005107                          ;       CALL    RTDSL           ; GET DRIVE STATE BEFORE INIT
005107  020000  001453          ;       ^O20000,RTDSL
005111  104010  002755          ;       MOV     R1,HIBN         ; SAVE IN UNUSED LOCATION
005113  104652  000025          ;       MOV     U.MASK(R5),R2   ; GET PORT MASK
005115  060011                  ;       XFC     DINIT           ; INIT THE DRIVE
005116                          ;       CALL    GOSEEK          ; SEEK REQUIRED IF INITED
005116  020000  005066          ;       ^O20000,GOSEEK
005120  101203  040000          ;       BIS     #DRINIT,R3      ; MARK THE DRIVE AS INITIALIZED
005122  100653  000047          ;       MOV     R3,U.RCOV(R5)   ; SAVE
005124  104201  001400          ;       MOV     #1400,R1        ; MOVE TIMEOUT TO R1
005126  117401                  ; 1$:   DEC     R1              ; DECREMENT TIMEOUT (WAIT FOR DRIVE CLOCKS)
005127                          ;       BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
005127  050000  005126          ;       ^O50000,1$
005131                          ;       POP     R1              ; RESTORE R1
005131  104261                                                                 MOV (SP)+,R1
005132                          ;       RETURN                  ; RETURN
005132  000000  000000          ;       ^O0,0
```

```
                                .SBTTL  ********** OVERLAY MODULE SETUP - OPERATION TO BE DONE THIS PASS
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
005134                          SETUP:
                                ;
                                ;       SETUP CLEARS ALL ERROR BITS, THE ERROR COUNT, FINDS THE NEXT
                                ;       LBN TO BE READ OR WRITTEN, DESIDES TO READ OR WRITE, CALCULATES
                                ;       THE CYLINDER, TRACK AND GROUP THE LBN RESIDES ON; THEN CALLS
                                ;       SEEK.
                                ;
                                        .ENABL  LSB
005134  114002                  CLR     R2                      ; NO ERRORS
005135  104657  000024          MOV     U.CSEC(R5),R0           ; GET TOTAL NUMBER OF SECTORS ALLREADY R/W
005137  105657  000021          ADD     U.NSEC(R5),R0           ; ADD HOW MANY R/W THIS PASS
005141  106657  000023          CMP     U.TSEC(R5),R0           ; SEE IF ALL DONE
005143                          BEQ     2$                      ; IF SO, BRANCH (NEW OPERATION)
005143  010000  005166          ^O010000,2$
005145  100657  000024          MOV     R0,U.CSEC(R5)           ; SAVE CURRENT COUNT
005147  104657  000053          MOV     U.CBN(R5),R0            ; GET LBN
005151  105657  000021          ADD     U.NSEC(R5),R0           ; ADD NUMBER OF SECTORS R/W THIS PASS
005153  100657  000053          MOV     R0,U.CBN(R5)            ; SAVE
005155                          BCC     9$                      ; IF NO CARRY, BRANCH
005155  040000  005245          ^O040000,9$
005157  104657  000054          MOV     U.CBN+1(R5),R0          ; GET HI LBN
005161  115407                  INC     R0                      ; PROPOGATE CARRY
005162  100657  000054          MOV     R0,U.CBN+1(R5)          ; SAVE
005164                          BR      9$                      ; BRANCH
005164  000000  005245          ^O00,9$
005166  104657  000046  2$:     MOV     U.PARM(R5),R0           ; GET UNIT PARAMETERS
005170  102207  000010          BIT     #WCHECK,R0              ; SEE IF WRITE CHECK IS TO BE DONE
005172                          BNE     5$                      ; IF SO, BRANCH
005172  050000  005226          ^O050000,5$
005174  104141                  MOV     (R4),R1                 ; GET SUBUNIT PARAMETERS
005175                          ASSUME  S.PARM,0                ; ASSUME THAT S.PARM IS ZERO
005175  102201  000100          BIT     #SEQSEK,R1              ; SEE IF SEQUENTIAL SEEKS
005177                          BNE     6$                      ; IF SO, BRANCH
005177  050000  005211          ^O050000,6$
005201  102207  040000          BIT     #INITW,R0               ; SEE IF INITIAL WRITE IN PROGRESS
005203                          BNE     6$                      ; IF SO, BRANCH
005203  050000  005211          ^O050000,6$
005205  101207  004000          BIS     #NEWSUB,R0              ; FLAG AS TO GO ONTO A NEW SUBUNIT
005207  100657  000046          MOV     R0,U.PARM(R5)           ; SAVE
005211  104207  005344  6$:     MOV     #NEWOP,R0               ; SET UP A NEW OPERATION
005213  114000  003002          CLR     SCR1                    ; CLEAR SCRATCH AREA1 (NO NEW SUBUNIT FLAG)
005215  104651  000023          MOV     U.TSEC(R5),R1           ; GET TOTAL NUMBER OF SECTORS TO BE WRIITEN LAST OP
005217                          BNE     4$                      ; IF NOT FIRST TIME, BRANCH
005217  050000  005341          ^O050000,4$
005221  104200  177777  003002  MOV     #-1,SCR1                ; FLAG AS NEW SUBUNIT (WITHOUT GOING ONTO NEW SUBUNIT)
005224                          BR      4$                      ; EXIT
005224  000000  005341          ^O00,4$
005226  103207  000110  5$:     BIC     #WCHECK!REDWRT,R0               ; CLEAR WRITE CHECK, MAKE OPERATION READ
005230  100657  000046          MOV     R0,U.PARM(R5)           ; SAVE
```

```
     84 005232  104657  000051                  MOV     U.MBN(R5),R0    ; GET LO LBN
     85 005234  100657  000053                  MOV     R0,U.CBN(R5)    ; SAVE
     86 005236  104657  000052                  MOV     U.MBN+1(R5),R0  ; GET HI LBN
     87 005240  100657  000054                  MOV     R0,U.CBN+1(R5)  ; SAVE
     88 005242  114002                          CLR     R2              ; NO ERRORS
     89 005243  100652  000024                  MOV     R2,U.CSEC(R5)   ; SECTORS R/W IS ZERO
     90 005245  104657  000023          9$:     MOV     U.TSEC(R5),R0   ; GET TOTAL NUMBER OF SECTORS TO R/W
     91 005247  107657  000024                  SUB     U.CSEC(R5),R0   ; SUBTRACT HOW MANY DONE
     92 005251  104651  000046                  MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
     93 005253  102201  000200                  BIT     #RBNBN,R1       ; SEE IF AN RBN IS BEING HANDLED
     94 005255                                  BEQ     1$              ; IF NOT, BRANCH
        005255  010000  005263                  ^O10000,1$
     95 005257  104207  000001                  MOV     #1,R0           ; ONLY READ/WRITE ONE SECTOR
     96 005261                                  BR      8$              ; BRANCH
        005261  000000  005305                  ^O00,8$
     97 005263  102201  000100          1$:     BIT     #REDWRT,R1      ; SEE IF WRITE IS IN PROGRESS
     98 005265                                  BNE     7$              ; IF SO, BRANCH
        005265  050000  005277                  ^O50000,7$
     99 005267  106307  003016                  CMP     SECMAX,R0       ; SEE IF TRYING TO READ MORE THAN POSSIBLE
    100 005271                                  BCC     8$              ; IF NOT, BRANCH
        005271  040000  005305                  ^O40000,8$
    101 005273  104307  003016                  MOV     SECMAX,R0       ; SET R0 TO MAXIMUM POSSIBLE SECTORS TO READ
    102 005275                                  BR      8$              ; BRANCH
        005275  000000  005305                  ^O00,8$
    103 005277  106307  003017          7$:     CMP     CHNMAX,R0       ; SEE IF TRYING TO WRITE MORE THAN POSSIBLE
    104 005301                                  BCC     8$              ; IF NOT, BRANCH
        005301  040000  005305                  ^O40000,8$
    105 005303  104307  003017                  MOV     CHNMAX,R0       ; SET R0 TO MAXIMUM POSSIBLE SECTORS TO WRITE
    106 005305  100657  000022          8$:     MOV     R0,U.MSEC(R5)   ; SAVE IN NUMBER OF SECTORS TO R/W THIS PASS
    107 005307  104653  000031          3$:     MOV     U.MLEV(R5),R3   ; GET MAXIMUM ERROR REVCOVERY LEVEL
    108 005311  100653  000027                  MOV     R3,U.ELEV(R5)   ; STORE IN CURRENT LEVEL
    109 005313  104203  177777                  MOV     #-1,R3          ; START RETRIES AT 0
    110 005315  100653  000012                  MOV     R3,U.RWTO(R5)   ; SAVE IN UNIT PARAMETER AREA
    111                                 ;       NOTE:   R2 IS ZERO AT THIS POINT
    112 005317  100652  000021                  MOV     R2,U.NSEC(R5)   ; NO SECTORES SUCCESSFULLY READ/WRITTEN
    113 005321  100652  000010                  MOV     R2,U.RRTY(R5)   ; ZERO READ RETRY VALUE
    114 005323  104207  015642                  MOV     #SEEK,R0        ; POINT TO SEEK AS NEXT OPERATION
    115 005325                                  CALL    ENABLE          ; ENABLE ERROR RECOVERY
        005325  020000  005036                  ^O20000,ENABLE
    116 005327  102203  030000                  BIT     #LEVUSD!NXTLEV,R3       ; SEE IF ERROR RECOVERY LEVELS USED
    117 005331                                  BEQ     4$              ; IF NOT, BRANCH
        005331  010000  005341                  ^O10000,4$
    118 005333                                  CALL    GOSEEK          ; CAUSE A SEEK TO RESET ERROR RECOVERY
        005333  020000  005066                  ^O20000,GOSEEK
    119 005335  103203  030000                  BIC     #LEVUSD!NXTLEV,R3       ; NO ERROR RECOVERY IN PROGRESS
    120 005337  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
    121 005341  114001                  4$:     CLR     R1              ; IMMIDIATE CALL
    122 005342                                  BR      JMPRET          ; RETURN TO SEQUENCER
        005342  000000  003746                  ^O00,JMPRET
    123                                         .DSABL  LSB
```

```
                                .SBTTL  ********** OVERLAY MODULE NEWOP - SET UP NEW OPERATION (COMMON CODE TOO)
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                ;
005344                          NEWOP:
                                ;
                                ;       SET UP A NEW OPERATION
                                ;
                                        .ENABL  LSB
005344  104147                         MOV     (R4),R0         ; GET SUBUNIT PARAMETERS
005345                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
005345  102207  040100                 BIT     #SEQSEK!INITW,R0        ; SEE IF BN'S ACCESSED SREQUENTIALLY
005347                                  BNE     2$              ; IF SO, BRANCH
005347  050000  005365                 ^O50000,2$
005351  102207  000040                 BIT     #BEUSED,R0      ; SEE IF BEGIN/END SETS USED
005353                                  BEQ     1$              ; IF NOT, BRANCH
005353  010000  005361                 ^O10000,1$
005355  104207  005650                 MOV     #RNDBE,R0       ; GET A RANDOM BLOCK NUMBER (BEGIN/END SETS)
005357                                  BR      4$              ; EXIT
005357  000000  005377                 ^O00,4$
005361  104207  006737         1$:     MOV     #RNDTG,R0       ; GET A RANDOM BLOCK NUMBER (TRACKS/GROUPS)
005363                                  BR      4$              ; BRANCH
005363  000000  005377                 ^O00,4$
005365  102207  000040         2$:     BIT     #BEUSED,R0      ; SEE IF BEGIN/END SETS USED
005367                                  BEQ     3$              ; IF NOT, BRANCH
005367  010000  005375                 ^O10000,3$
005371  104207  006060                 MOV     #SEQBE,R0       ; GET A SEQUENTIAL BLOCK NUMBER (BEGIN/END SETS)
005373                                  BR      4$              ; EXIT (SHOULD BRANCH TO 7$) ********
005373  000000  005377                 ^O00,4$
005375  104207  007147         3$:     MOV     #SEQTG,R0       ; GET A SEQUENTIAL BLOCK NUMBER (TRACKS/GROUPS)
005377  114001                 4$:     CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
005400  114002                         CLR     R2              ; NO ERRORS
005401                                  BR      JMPRET
005401  000000  003746                 ^O00,JMPRET
                                        .DSABL  LSB
```

```
                                        .SBTTL  AFTOP  - AFTER THE SECTOR HAS BEEN DETERMINED, FIND OUT WHAT TO DO WITH IT
005403                                  AFTOP:
                                        ;
                                        ;       AFTER THE SECTORS TO BE READ/WRITTEN HAVE BEEN DETERMINED, SET UP THE
                                        ;       REST OF THE OPERATION
                                        ;
                                        .SBTTL  CLRUP  - CLEAR ALL PARAMETER BITS
                                        ;CLRUP
                                        ;
                                        ;       CLRUP CLEARS ALL NECESSARY FLAG BITS
                                        ;
005403  104657  000046                  ;       MOV     U.PARM(R5),R0   ; MOVE UNIT PARAMETERS TO R0
                                        ;       CLEAR ALL FLAGS BEFORE THE NEXT OPERATION
005405  103207  004712                  ;       BIC     #RBNBN!REVEC!WCHECK!DATCMP!REDWRT!NEWSUB,R0
005407  100657  000046                  ;       MOV     R0,U.PARM(R5)   ; SAVE UNIT PARAMETERS
```

```
      1                                 .SBTTL  RORW   - DETERMINE IF A READ OR A WRITE IS TO BE DONE
      2                                 ;RORW
      3                                 ;
      4                                 ;       RORW DETERMINES IF THE BLOCK SELECTED WILL BE READ OR WRITTEN
      5                                 ;
      6 005411  104142                          MOV     (R4),R2         ; GET SUBUNIT PARAMETERS
      7 005412                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
      8 005412  102202  042000                  BIT     #INITW!WONLY,R2 ; SEE IF INITIAL WRITE OR WRITE ONLY
      9 005414                                  BNE     1$              ; IF SO, BRANCH
        005414  050000  005427                  ^O50000,1$
     10 005416  102202  004000                  BIT     #RONLY,R2       ; SEE IF READ ONLY
     11 005420                                  BNE     2$              ; IF SO, BRANCH
        005420  050000  005433                  ^O50000,2$
     12 005422                                  CALL    RANDOM          ; GET RANDOM NUMBER
        005422  020000  005563                  ^O20000,RANDOM
     13 005424  110601                          ROR     R1              ; ROTATE LOW BIT INTO CARRY
     14 005425                                  BCC     2$              ; IF LOW BIT ZERO, BRANCH
        005425  040000  005433                  ^O40000,2$
     15 005427  101207  000100          1$:     BIS     #REDWRT,R0      ; SET READ OR WRITE BIT (WRITE)
     16 005431  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE UNIT PARAMETERS
     17 005433                          2$:
```

```
      1                                 .SBTTL  PATTRN - IF WRITE, DETERMINE WHAT PATTERN IS TO BE WRITTEN
      2                                 ;PATTRN
      3                                 ;
      4                                 ;       FIND PATTERN TO USE (ONLY IF WRITE TO BE DONE)
      5                                 ;
      6 005433  102207  000100                   BIT     #REDWRT,R0      ; SEE IF WRITE TO BE DONE
      7 005435                                    BEQ     PATEXT          ; IF NOT, BRANCH
        005435  010000  005457                    ^010000,PATEXT
      8 005437  104641  000004                    MOV     S.PAT(R4),R1    ; GET PATTERN NUMBER
      9 005441                                    BEQ     RND0            ; IF PATTERN NO' SELECTED, BRANCH
        005441  010000  005447                    ^010000,RND0
     10 005443  103201  177760                    BIC     #LBLONB,R1      ; SPECIAL PATTERN USED, CLEAR HI BITS (MAP 16 TO 0)
     11 005445                                    BR      FIXPAT          ; IF PATTERN SELECTED, USE IT
        005445  000000  005455                    ^00,FIXPAT
     12 005447                          RND0:    CALL    RANDOM          ; GET RANDOM PATTERN NUMBER
        005447  020000  005563                    ^020000,RANDOM
     13 005451  103201  177760                    BIC     #LBLONB,R1      ; CLEAR UNUSED BITS
     14 005453                                    BEQ     RND0            ; NO SUCH THING AS PATTERN 0, TRY AGAIN
        005453  010000  005447                    ^010000,RND0
     15 005455  100651  000014          FIXPAT:  MOV     R1,U.PAT(R5)    ; SAVE THE PATTERN NUMBER
     16 005457                          PATEXT:
```

```
                                .SBTTL  WCHK   - IF WRITE, SEE IF A WRITE CHECK IS TO BE DONE
                                ;WCHK
                                ;
                                ;       WCHK SETS THE WRITE CHECK BIT IF A WRITE CHECK IS TO BE DONE
                                ;
005457  102207  000100                  BIT     #REDWRT,R0      ; SEE IF WRITE IS TO BE DONE
005461                                  BEQ     2$              ; IF NOT, BRANCH
005461  010000  005510                  ^O10000,2$
005463  102202  000010                  BIT     #WCHECK,R2      ; SEE IF WRITE CHECK IS TO BE DONE
005465                                  BEQ     2$              ; IF NOT, BRANCH
005465  010000  005510                  ^O10000,2$
005467  102202  040000                  BIT     #INITW,R2       ; SEE IF INITIAL WRITE IS IN PROGRESS
005471                                  BNE     2$              ; IF SO, NO WRITE CHECK, SO BRANCH
005471  050000  005510                  ^O50000,2$
005473  102202  000004                  BIT     #WCHKAL,R2      ; SEE IF WRITE CHECK IS ALWAYS TO BE DONE
005475                                  BNE     1$              ; IF SO, BRANCH
005475  050000  005504                  ^O50000,1$
005477                                  CALL    RANDOM          ; GET RANDOM NUMBER
005477  020000  005563                  ^O20000,RANDOM
005501  110601                          ROR     R1              ; 1/2 OF THE TIME WRITE CHECK
005502                                  BCC     2$              ; BRANCH (NO WRITE CHECK) IF LOWEST BIT CLEAR
005502  040000  005510                  ^O40000,2$
005504  101207  000010          1$:     BIS     #WCHECK,R0      ; SET WRITE CHECK BIT
005506  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE UNIT PARAMETERS
005510                          2$:
```

```
                                        .SBTTL  DCOMP  - IF READ OR WRITE W/WRITE CHECK, SEE IF DATA COMPARE IS TO BE DONE
                                        ;DCOMP
                                        ;
                                        ;       DCOMP SETS THE DATCMP BIT IF A DATA COMPARE IS TO BE DONE
                                        ;
005510  102202  000002                  BIT     #DATCMP,R2      ; SEE IF DATA COMPARE REQUESTED
005512                                  BEQ     DCEXT           ; IF NOT, BRANCH
005512  010000  005541                  ^O10000,DCEXT
005514  102207  000100                  BIT     #REDWRT,R0      ; SEE IF READ IS TO BE DONE
005516                                  BEQ     DCREAD          ; IF SO, BRANCH
005516  010000  005524                  ^O10000,DCREAD
005520  102207  000010                  BIT     #WCHECK,R0      ; SEE IF WRITE CHECK IS TO BE DONE
005522                                  BEQ     DCEXT           ; IF NOT, BRANCH
005522  010000  005541                  ^O10000,DCEXT
005524  102202  000001          DCREAD: BIT     #DCMPAL,R2      ; SEE IF DATA COMPARE ALWAYS DONE
005526                                  BNE     DCMPYS          ; IF SO, BRANCH
005526  050000  005535                  ^O50000,DCMPYS
005530                                  CALL    RANDOM          ; GET RANDOM NUMBER
005530  020000  005563                  ^O20000,RANDOM
005532  110601                          ROR     R1              ; SEE IF DATA COMPARE SHOULD BE DONE
005533                                  BCC     DCEXT           ; IF NOT, BRANCH
005533  040000  005541                  ^O40000,DCEXT
005535  101207  000002          DCMPYS: BIS     #DATCMP,R0      ; SET DATA COMPARE BIT
005537  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE UNIT PARAMETERS
005541                          DCEXT:
```

```
      1 005541  104657  000051                MOV     U.MBN(R5),R0    ; GET LO LBN
      2 005543  100657  000053                MOV     R0,U.CBN(R5)    ; SAVE
      3 005545  104657  000052                MOV     U.MBN+1(R5),R0  ; GET HI LBN
      4 005547  100657  000054                MOV     R0,U.CBN+1(R5)  ; SAVE
      5 005551  114002                        CLR     R2              ; NO ERRORS
      6 005552  100652  000024                MOV     R2,U.CSEC(R5)   ; SECTORS R/W IS ZERO
      7 005554  100652  000021                MOV     R2,U.NSEC(R5)   ; SECTORS R/W THIS PASS IS ZERO
      8 005556  114001                        CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
      9 005557  104207  005134                MOV     #SETUP,R0       ; NEXT MODULE IS SETUP
     10 005561                                BR      JMPRET          ; RETURN TO SEQUENCER
        005561  000000  003746                ^O0,JMPRET
```

```
                                        .SBTTL  RANDOM - RANDOM NUMBER ROUTINE
005563                                  RANDOM:
                                        ;
                                        ;       RANDOM CALCULATES A RANDOM NUMBER AND RETURNS IT IN R1
                                        ;       RANGE 0 - 2^16-1
                                        ;
005563                                          PUSH    <R0,R2>         ; SAVE R0 AND R2
005563  100467                                                                  MOV R0,-(SP)
005564  100462                                                                  MOV R2,-(SP)
005565  104307  003007                          MOV     LOSEED,R0       ; MOVE LO ORDER SEED TO R0
005567  104301  003010                          MOV     HISEED,R1       ; MOVE HI SEED TO R1
005571  104202  000007                          MOV     #7,R2           ; MOVE LOOP COUNT TO R2
005573  110207                          1$:     ROL     R0              ; ROTATE LO ORDER NUMBER BY 1
005574  110201                                  ROL     R1              ; ROTATE HI OREDR NUMBER BY 1 (PROPOGATE CARRY)
005575  103207  000001                          BIC     #1,R0           ; CLEAR LOWER BIT
005577  117402                                  DEC     R2              ; DECREMENT COUNT
005600                                          BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
005600  050000  005573                          ^050000,1$
005602  105307  003007                          ADD     LOSEED,R0       ; ADD ORIGINAL SEED (X129)
005604                                          BCC     2$              ; IF NO CARRY, BRANCH
005604  040000  005607                          ^040000,2$
005606  115401                                  INC     R1              ; PROPOGATE CARRY
005607  105301  003010                  2$:     ADD     HISEED,R1       ; ADD HISEED
005611  105207  001057                          ADD     #1057,R0        ; ADD LO CONSTANT
005613                                          BCC     3$              ; IF NO CARRY, BRANCH
005613  040000  005616                          ^040000,3$
005615  115401                                  INC     R1              ; PROPOGATE CARRY
005616  105201  047401                  3$:     ADD     #47401,R1       ; ADD HI CONSTANT
005620  104070  003007                          MOV     R0,LOSEED       ; SAVE LO ORDER SEED
005622  104010  003010                          MOV     R1,HISEED       ; SAVE HI ORDER SEED
005624                                          POP     <R2,R0>         ; RESTORE R2 AND R0
005624  104262                                                                  MOV (SP)+,R2
005625  104267                                                                  MOV (SP)+,R0
005626                                          RETURN
005626  000000  000000                          ^00,0
```

```
                                .SBTTL  MASK    - FIND MASK FOR RANDOM NUMBER
005630                          MASK:
                                ;
                                ;       MASK MASKS OUT HIGHER BITS OF A RANDOM NUMBER, SO THE PROB. OF THE
                                ;       RANDOM NUMBER EXCEEDING THE MAXIMUM DESIRED IS LESSENED
                                ;
005630                          PUSH    R1              ; SAVE R1
005630  100461                                                          MOV R1,-(SP)
005631  114001                  CLR     R1              ; ZERO R1
005632  105011          1$:     ADD     R1,R1           ; ROTATE R1 ONE BIT LEFT
005633  101201  000001         BIS     #1,R1           ; TURN ON BIT 0
005635  106017                  CMP     R1,R0           ; SEE IF R1 IS GREATER THAN R0
005636                          BCS     1$              ; IF NOT, BRANCH
005636  040000  005642         ^040000,..+3
005640  000000  005632         ^00,1$
005642  104207  177777         MOV     #-1,R0          ; SET UP FOR COMPLEMENT
005644  103017                  BIC     R1,R0           ; COMPLEMENT R1
005645                          POP     R1              ; RESTORE R1
005645  104261                                                          MOV (SP)+,R1
005646                          RETURN                  ; RETURN TO RNDLBN
005646  000000  000000         ^00,0
```

```
                                .SBTTL  ********** OVERLAY MODULE RNDBE - GET NEXT RANDOM BEGIN/END SET SECTOR
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                ;
005650                          RNDBE:
                                ;
                                ;       RNDBE GETS A RANDOM LBN NUMBER
                                ;
                                ;       GET A RANDOM BEGIN/END SET FROM THE SUBUNIT'S B/E SETS
                                ;
                                        .ENABL  LSB
005650  114007                  CNTLOP: CLR     R0              ; INITILIZE COUNTER
005651  104203  000016                  MOV     #S.BESS+3,R3    ; R3 POINTS TO FIRST B/E SET
005653  105043                          ADD     R4,R3           ; R3 POINTS TO FIRST B/E SET
005654  104132                          MOV     (R3),R2         ; MOVE EOL FLAG WORD TO R2
005655                                  BMI     COUNTD          ; IF END-OF-LIST, BRANCH
005655  070000  005664                  ^070000,COUNTD
005657  105203  000004                  ADD     #4,R3           ; POINT TO NEXT EOL WORD
005661  115407                          INC     R0              ; INCREMENT COUNT
005662                                  BR      CNTLOP          ; IF NOT, TRY AGAIN
005662  000000  005654                  ^00,CNTLOP
005664  115007                  COUNTD: TST     R0              ; SEE IF COUNT = 0
005665                                  BEQ     GOTOFS          ; IF SO, BRANCH
005665  010000  005714                  ^010000,GOTOFS
005667                          TRYAGN: CALL    RANDOM          ; GET A RANDOM NUMBER
005667  020000  005563                  ^020000,RANDOM
005671  103201  177774                  BIC     #177774,R1      ; MASK OUT ALL BUT 2 LOWEST BITS
005673  106207  000002                  CMP     #2,R0           ; TEST HOW COUNT RELATES TO 2
005675                                  BMI     GOTOBE          ; IF COUNT IS 3, BRANCH
005675  070000  005711                  ^070000,GOTOBE
005677                                  BEQ     THREBE          ; IF COUNT IS 2, BRANCH
005677  010000  005705                  ^010000,THREBE
005701  103201  177776                  BIC     #177776,R1      ; MASK OUT ALL BUT LOWEST BIT
005703                                  BR      GOTOBE          ; BRANCH
005703  000000  005711                  ^00,GOTOBE
005705  106201  000002          THREBE: CMP     #2,R1           ; SEE IF RANDOM NUMBER OVER 2
005707                                  BMI     TRYAGN          ; IF SO, GET ANOTHER NUMBER
005707  070000  005667                  ^070000,TRYAGN
005711  104017                  GOTOBE: MOV     R1,R0           ; MOVE RANDOM NUMBER TO R0
005712  105077                          ADD     R0,R0           ; R0 X 2
005713  105077                          ADD     R0,R0           ; R0 X 2
005714  105207  000013          GOTOFS: ADD     #S.BESS,R0      ; POINT TO RANDOM BEGIN/END SET
005716  105047                          ADD     R4,R0           ; POINT TO RANDOM BEGIN/END SET
```

```
                                        ;
                                        ;       ONCE A RANDOM B/E SET HAS BEEN FOUND, FIND A RANDOM LBN
                                        ;       WITHIN THAT B/E SET
                                        ;
 5 005717                               PUSH    <R4,R5>         ; SAVE R4 AND R5
   005717  100464                                                                  MOV R4,-(SP)
   005720  100465                                                                  MOV R5,-(SP)
 6 005721  104274                       MOV     (R0)+,R4        ; R4 GETS LO ORDER ENDING LBN
 7 005722  104275                       MOV     (R0)+,R5        ; R5 GETS HI ORDER ENDING LBN
 8 005723  107274                       SUB     (R0)+,R4        ; SUBTRACT LO ORDER STARTING LBN
 9 005724                               BCC     NOBOR0          ; IF NO BORROW, BRANCH
   005724  040000  005727               ^O040000,NOBOR0
10 005726  117405                       DEC     R5              ; ADJUST FOR BORROW
11 005727  104171                NOBOR0: MOV     (R0),R1         ; GET HI WORD
12 005730  103201  170000               BIC     #170000,R1      ; STRIP OFF EOL FLAG (IF ANY)
13 005732  107015                       SUB     R1,R5           ; SUBTRACT HI ORDER STARTING LBN
14 005733  117407                       DEC     R0              ; POINT TO LO ORDER STARTING LBN
15 005734                               PUSH    R0              ; SAVE R0
   005734  100467                                                                  MOV R0,-(SP)
16 005735                        10$:   CALL    RANDOM          ; GET A RANDOM NUMBER
   005735  020000  005563               ^O020000,RANDOM
17 005737  114003                       CLR     R3              ;
18 005740  104301  003007               MOV     LOSEED,R1       ; ORIGINAL SEED IN R1
19 005742  115005                       TST     R5              ; SEE IF R5 IS ZERO
20 005743                               BEQ     11$             ; IF SO, BRANCH
   005743  010000  005762               ^O010000,11$
21 005745  104057                       MOV     R5,R0           ; ELSE, STORE IN R0
22 005746                               CALL    MASK            ; COMPUTE RANDOM NUMBER MASK
   005746  020000  005630               ^O020000,MASK
23 005750  104303  003010               MOV     HISEED,R3       ; MOVE RANDOM NUMBER INTO R3
24 005752  103073                       BIC     R0,R3           ; CLEAR WITH MASK
25 005753  106053                       CMP     R5,R3           ; CMP MAXIMUM RANDOM NUM TO RANDOM NUM
26 005754                               BCS     10$             ; IF TOO LARGE, DO AGAIN
   005754  040000  005760               ^O040000,.+3
   005756  000000  005735               ^O0,10$
27 005760                               BNE     CLCLBN          ; IF NOT THE SAME, BRANCH
   005760  050000  005773               ^O050000,CLCLBN
28 005762  104047                11$:   MOV     R4,R0           ; MOVE R4 TO R0 TO COMPUTE MASK
29 005763                               CALL    MASK            ; COMPUTE MASK
   005763  020000  005630               ^O020000,MASK
30 005765  103071                       BIC     R0,R1           ; MASK OUT HIGH BITS
31 005766  106041                       CMP     R4,R1           ; SEE IF MAX RND NUM < RANDOM NUMBER
32 005767                               BCS     10$             ; IF SO, DO AGAIN
   005767  040000  005773               ^O040000,.+3
   005771  000000  005735               ^O0,10$
33 005773  104012                CLCLBN: MOV     R1,R2           ; MOVE RANDOM NUMBER TO R2
34 005774                               POP     R0              ; RESTORE R0
   005774  104267                                                                 MOV (SP)+,R0
35 005775  105272                       ADD     (R0)+,R2        ; ADD LO ORDER STARTING BLOCK TO RANDOM NUM
36 005776                               BCC     3$              ; IF NO CARRY, BRANCH
   005776  040000  006001               ^O040000,3$
37 006000  115403                       INC     R3              ; ADJUST HIGH ORDER FOR CARRY
38 006001  104171                3$:    MOV     (R0),R1         ; MOVE HI ORDER LBN TO R1
39 006002  103201  170000               BIC     #170000,R1      ; STRIP OFF EOL FLAG IF ANY
40 006004  105013                       ADD     R1,R3           ; ADD HI ORDER LBN
41 006005                               POP     <R5,R4>         ; RESTORE R5, R4
   006005  104265                                                                 MOV (SP)+,R5
```

```
                006006  104264                                                          MOV (SP)+,R4
 42             006007  107207  000003          SUB     #3,R0           ; R0 POINTS TO ENDING LBN
 43             006011  104271                  MOV     (R0)+,R1        ; R1 HAS LO ORDER ENDING LBN
 44             006012  104177                  MOV     (R0),R0         ; R0 HAS HI ORDER ENDING LBN
 45             006013  107021                  SUB     R2,R1           ; SUBTRACT LO LBN FROM LO ENDING LBN
 46             006014                          BMI     4$              ; IF RESULT IS NEGATIVE, BRANCH
                006014  070000  006024          ^070000,4$
 47             006016                          BCC     1$              ; IF NO CARRY, BRANCH
                006016  040000  006021          ^040000,1$
 48             006020  117407                  DEC     R0              ; DECREMENT HI ENDING LBN
 49             006021  107037          1$:     SUB     R3,R0           ; SUBTRACT HI LBN FROM HI ENDING LBN
 50             006022                          BEQ     2$              ; IF HI WORD IS ZERO, BRANCH
                006022  010000  006026          ^010000,2$
 51             006024  104201  077776  4$:     MOV     #77776,R1       ; MOVE MAXIMUM COUNT TO R1
 52             006026  115401          2$:     INC     R1              ; INCREMENT MAXIMUM SECTOR COUNT
 53             006027  104017                  MOV     R1,R0           ; COPY TO R0 FOR MAXMUM
 54
 55             006030                          PUSH    R3              ; SAVE R3
                006030  100463                                                          MOV R3,-(SP)
 56             006031                          CALL    RANDOM          ; GET A RANDOM NUMBER OF SECTORS TO READ/WRITE
                006031  020000  005563          ^020000,RANDOM
 57             006033  103201  177740          BIC     #MAXMSK,R1      ; SET READ/WRITE LIMIT
 58             006035  115401                  INC     R1              ; MAKE MAXIMUM OF LIMIT+1
 59             006036  106071                  CMP     R0,R1           ; SEE IF RANDOM NUMBER EXCEEDS POSSIBLE
 60             006037                          BMI     6$              ; IF SO, BRANCH
                006037  070000  006042          ^070000,6$
 61             006041  104017                  MOV     R1,R0           ; ONLY READ/WRITE THE RANDOM NUMBER OF SECTORS
 62             006042  100657  000023  6$:     MOV     R0,U.TSEC(R5)   ; SAVE IN TSEC
 63             006044                          POP     R3              ; RESTORE R3
                006044  104263                                                          MOV (SP)+,R3
 64             006045  104207  000051          MOV     #U.MBN,R0       ; R0 POINTS TO LBN AREA
 65             006047  105057                  ADD     R5,R0           ; R0 POINTS TO LBN AREA
 66             006050  100272                  MOV     R2,(R0)+        ; MOVE LO ORDER TO LBN AREA
 67             006051  100173                  MOV     R3,(R0)         ; MOVE HI ORDER TO LBN AREA
 68             006052  104207  005403          MOV     #AFTOP,R0       ; NEXT MODULE IS AFTOP
 69             006054  114001                  CLR     R1              ; IMMIDATE CALL
 70             006055  114002                  CLR     R2              ; NO ERRORS
 71             006056                          BR      JMPRET
                006056  000000  003746          ^O0,JMPRET
 72                                              .DSABL  LSB
```

```
                                .SBTTL  ********** OVERLAY MODULE SEQBE - GET NEXT SEQUENTIAL BEGIN/END SET SECTOR
                                ;**************************************************************************************
                                ;**************************************************************************************
                                ;**************************************************************************************
                                ;**************************************************************************************
                                ;
                                ;
                                ;
006060                          SEQBE:
                                ;
                                ;       SEQBE FINDS THE NEXT SEQUENTIAL LBN
                                ;
                                        .ENABL  LSB
006060  104657  000046                  MOV     U.PARM(R5),R0   ; MOVE UNIT PARAMETERS TO R0
006062  102207  010000                  BIT     #DIREC,R0       ; TEST DIRECTION
006064                                  BEQ     UP              ; IF SEQUENTIAL UP, BRANCH
006064  010000  006212                  ^O10000,UP
006066  115000  003002                  TST     SCR1            ; SEE IF NEW SUBUNIT
006070                                  BEQ     5$              ; IF NOT, BRANCH
006070  010000  006117                  ^O10000,5$
006072  104201  000013                  MOV     #S.BESS,R1      ; R0 WILL POINT TO BEGIN/END SETS
006074  105041                          ADD     R4,R1           ; R0 POINTS TO BEGIN/END SETS
006075  104617  000003          6$:     MOV     3(R1),R0        ; SEE IF END-OF-LIST
006077                                  BMI     8$              ; IF SO, BRANCH
006077  070000  006105                  ^O70000,8$
006101  105201  000004                  ADD     #4,R1           ; POINT TO NEXT BEGIN/END SET
006103                                  BR      6$              ; LOOP
006103  000000  006075                  ^O00,6$
006105  104202  000051          8$:     MOV     #U.MBN,R2       ; R2 WILL POINT TO MASTER BN
006107  105052                          ADD     R5,R2           ; R2 POINTS TO MASTER BN
006110  104217                          MOV     (R1)+,R0        ; GET LO ORDER ENDING SET
006111  100127                          MOV     R0,(R2)         ; SAVE
006112  104217                          MOV     (R1)+,R0        ; GET HI ORDER ENDING SET
006113  100627  000001                  MOV     R0,1(R2)        ; SAVE
006115                                  BR      7$              ; BRANCH
006115  000000  006142                  ^O00,7$
006117                          5$:     CALL    FNDBES          ; FIND BEGIN/END SET THAT LBN IS IN
006117  020000  006343                  ^O20000,FNDBES
006121  105201  000002                  ADD     #2,R1           ; POINT TO BEGIN SET
006123                                  CALL    CMP2            ; SEE IF LBN = BEGINNING LBN
006123  020000  002536                  ^O20000,CMP2
006125                                  BEQ     BESDWN          ; IF SO, BRANCH
006125  010000  006167                  ^O10000,BESDWN
006127  104127                          MOV     (R2),R0         ; MOVE LOW ORDER WORD TO R0
006130  107207  000001                  SUB     #1,R0           ; DECREMENT R0
006132  100127                          MOV     R0,(R2)         ; SAVE R0
006133                                  BCC     7$              ; IF NO BORROW, BRANCH
006133  040000  006142                  ^O40000,7$
006135  104627  000001                  MOV     1(R2),R0        ; MOVE HIGH ORDER WORD TO R0
006137  117407                          DEC     R0              ; DECREMENT R0
006140  100627  000001                  MOV     R0,1(R2)        ; SAVE R0
006142  104227                  7$:     MOV     (R2)+,R0        ; MOVE LO ORDER BN TO R0
006143  104223                          MOV     (R2)+,R3        ; MOVE HI ORDER BN TO R3
006144  107217                          SUB     (R1)+,R0        ; SUBTRACT BEGIN BN FROM BN
006145                                  BMI     4$              ; IF RESULT IS NEGATIVE, BRANCH
006145  070000  006160                  ^O70000,4$
006147                                  BCC     1$              ; IF NO BORROW, BRANCH
```

```
            006147  040000  006152                  ^040000,1$
         66 006151  117403                          DEC     R3              ; PROPOGATE BORROW
         67 006152  104111                  1$:     MOV     (R1),R1         ; GET BEGINNING BN
         68 006153  103201  170000                   BIC     #^CHBHINB,R1    ; CLEAR EOL FLAG, IF ANY
         69 006155  107013                          SUB     R1,R3           ; SUBTRACT HI ORDER BN FROM HI BN
         70 006156                                  BEQ     2$              ; IF EQUAL, BRANCH
            006156  010000  006162                  ^010000,2$
         71 006160  104207  077776          4$:     MOV     #77776,R0       ; MOVE MAXIMUM COUNT TO R0
         72 006162  115407                  2$:     INC     R0              ; MAKE SUBTRACTION INCLUSIVE
         73 006163                          3$:     CALL    MAXMUM          ; SET MAXIMUM COUNT (U.TSEC AND U.MSEC)
            006163  020000  006366                  ^020000,MAXMUM
         74 006165                                  BR      NLBEXT          ; BRANCH
            006165  000000  006171                  ^00,NLBEXT
         75                                         .DSABL  LSB
         76 006167                          BESDWN: CALL    PREVBE          ; FIND PREVIOUS B/E SET OR UNIT
            006167  020000  006414                  ^020000,PREVBE
         77 006171  104651  000023          NLBEXT: MOV     U.TSEC(R5),R1   ; GET TOTAL NUMBER OF SECTORES TO READ/WRITE
         78 006173  117401                          DEC     R1              ; DECREMENT SECTOR COUNT
         79 006174  104653  000051                  MOV     U.MBN(R5),R3    ; GET LO ORDER MASTER BN
         80 006176  107013                          SUB     R1,R3           ; SUBTRACT SECTORS TO READ/WRITE
         81 006177  100653  000051                  MOV     R3,U.MBN(R5)    ; SAVE
         82 006201                                  BCC     1$              ; IF NO CARRY, BRANCH
            006201  040000  006210                  ^040000,1$
         83 006203  104653  000052                  MOV     U.MBN+1(R5),R3  ; GET HI ORDER MASTER BN
         84 006205  117403                          DEC     R3              ; PROPOGATE CARRY
         85 006206  100653  000052                  MOV     R3,U.MBN+1(R5)  ; SAVE
         86 006210                          1$:     BR      BEEXT           ; BRANCH
            006210  000000  006323                  ^00,BEEXT
         87 006212  115000  003002          UP:     TST     SCR1            ; SEE IF FIRST TIME ON THIS SUBUNIT
         88 006214                                  BEQ     5$              ; IF NOT, BRANCH
            006214  010000  006237                  ^010000,5$
         89 006216  104201  000013                  MOV     #S.BESS,R1      ; R0 WILL POINT TO BEGIN/END SETS
         90 006220  105041                          ADD     R4,R1           ; R0 POINTS TO BEGIN/END SETS
         91 006221  104202  000051                  MOV     #U.MBN,R2       ; R2 WILL POINT TO MASTER BN
         92 006223  105052                          ADD     R5,R2           ; R2 POINTS TO MASTER BN
         93 006224  104617  000002                  MOV     2(R1),R0        ; GET LO ORDER STARTING SET
         94 006226  100127                          MOV     R0,(R2)         ; SAVE
         95 006227  104617  000003                  MOV     3(R1),R0        ; GET HI ORDER STARTING SET
         96 006231  103207  170000                  BIC     #^CHBHINB,R0    ; CLEAR EOL FLAG IF ANY
         97 006233  100627  000001                  MOV     R0,1(R2)        ; SAVE
         98 006235                                  BR      7$              ; BRANCH
            006235  000000  006277                  ^00,7$
         99 006237                          5$:
        100                                 .SBTTL  UPLBN  - IF GOING UP, UPDATE CURRENT BN TO LAST BN READ/WRITTEN
        101                                 ;UPLBN
        102                                 ;
        103                                 ;       UPLBN UPDATES U.MBN TO POINT TO THE LAST LBN READ/WRITTEN
        104                                 ;       (INCREASING LBN)
        105                                 ;
        106 006237                                  PUSH    <R0,R2>         ; SAVE R0,R2
            006237  100467                                                                  MOV R0,-(SP)
            006240  100462                                                                  MOV R2,-(SP)
        107 006241  104202  000051                  MOV     #U.MBN,R2       ; R2 POINTS TO LAST LBN
        108 006243  105052                          ADD     R5,R2           ; R2 POINTS TO LAST LBN
        109 006244  104657  000023                  MOV     U.TSEC(R5),R0   ; GET NUMBER OF SECTORS WRITTEN
        110 006246  117407                          DEC     R0              ; SUBTRACT ONE
        111 006247  105127                          ADD     (R2),R0         ; ADD LOW ORDER LBN TO R0
```

```
                                        MOV     R0,(R2)+        ; SAVE LOW ORDER WORD, POINT TO HIGH ORDER
                                        BCC     8$              ; IF NO CARRY, BRANCH
006251  040000  006256                  ^040000,8$
                                        MOV     (R2),R0         ; GET HIGH ORDER WORD
                                        INC     R0              ; INCREMENT
                                        MOV     R0,(R2)         ; SAVE HIGH ORDER WORD
                                8$:     POP     <R2,R0>         ; RESTORE R0,R2
006256  104262                                                                  MOV (SP)+,R2
006257  104267                                                                  MOV (SP)+,R0
                                        CALL    FNDBES          ; FIND BEGIN/END SET THAT LBN IS IN
006260  020000  006343                  ^020000,FNDBES
                                        BEQ     BESUP           ; IF LBN = ENDING LBN, BRANCH
006262  010000  006321                  ^010000,BESUP
                                        MOV     (R2),R0         ; MOVE LOW ORDER WORD TO R0
006264  105207  000001                  ADD     #1,R0           ; INCREMENT R0
006266  100127                          MOV     R0,(R2)         ; SAVE R0
                                        BCC     7$              ; IF NO CARRY, BRANCH
006270  040000  006277                  ^040000,7$
006272  104627  000001                  MOV     1(R2),R0        ; MOVE HIGH ORDER WORD TO R0
006274  115407                          INC     R0              ; INCREMENT R0
006275  100627  000001                  MOV     R0,1(R2)        ; SAVE R0
                                7$:     MOV     (R1)+,R0        ; R0 HAS LO ENDING LBN
                                        MOV     (R1),R3         ; R3 HAS HI ENDING LBN
                                        SUB     (R2)+,R0        ; SUBTRACT LO LBN FROM LO ENDING LBN
                                        BMI     4$              ; IF RESULT IS NEGATIVE, BRANCH
006302  070000  006312                  ^070000,4$
                                        BCC     1$              ; IF NO CARRY, BRANCH
006304  040000  006307                  ^040000,1$
                                        DEC     R3              ; PROPOGATE CARRY
                                1$:     SUB     (R2),R3         ; SUBTRACT HI LBN FROM ENDING LBN
                                        BEQ     2$              ; IF ZERO, BRANCH
006310  010000  006314                  ^010000,2$
006312  104207  077776          4$:     MOV     #77776,R0       ; MOVE MAXIMUM COUNT TO R0
006314  115407                  2$:     INC     R0              ; MAKE SUBTRACTION INCLUSIVE
                                3$:     CALL    MAXMUM          ; SET MAXIMUM COUNT (U.TSEC AND U.MSEC)
006315  020000  006366                  ^020000,MAXMUM
                                        BR      BEEXT           ; BRANCH
006317  000000  006323                  ^00,BEEXT
                                BESUP:  CALL    NEXTBE          ; FIND NEXT B/E SET OR UNIT
006321  020000  006460                  ^020000,NEXTBE
006323  104657  000046          BEEXT:  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
006325  102207  004000                  BIT     #NEWSUB,R0      ; SEE IF NEW SUBUNIT
                                        BEQ     1$              ; IF NOT, BRANCH
006327  010000  006335                  ^010000,1$
006331  104207  005344                  MOV     #NEWOP,R0       ; NEWOP NEXT MODULE
                                        BR      2$              ; BRANCH
006333  000000  006337                  ^00,2$
006335  104207  005403          1$:     MOV     #AFTOP,R0       ; NEXT MODULE IS AFTOP
006337  114001                  2$:     CLR     R1              ; IMMIDATE CALL
006340  114002                          CLR     R2              ; NO ERRORS
                                        BR      JMPRET          ; RETURN TO SEQUENCER
006341  000000  003746                  ^00,JMPRET
```

```
                                        .SBTTL  FNDBES - FIND THE BEGIN/END SET CURRENT BN RESIDES IN
006343                                  FNDBES:
                                        ;
                                        ;       FNDBES FINDS THE BEGIN/END SET THAT THE CURRENT LBN RESIDES IN
                                        ;
006343  104203  177777                          MOV     #-1,R3          ; START COUNTER (ADJUSTED)
006345  104201  000007                          MOV     #S.BESS-4,R1    ; POINT TO B/E SETS
006347  105041                                  ADD     R4,R1           ; POINT TO B/E SETS
006350  104202  000051                          MOV     #U.MBN,R2       ; POINT TO LBN
006352  105052                                  ADD     R5,R2           ; POINT TO LBN
006353  105201  000004                FNDLOP:   ADD     #4,R1           ; INCREMENT B/E SET POINTER
006355  115403                                  INC     R3              ; INCREMENT COUNT
006356                                          CALL    CMP2            ; COMPARE LBN WITH ENDING LBN
006356  020000  002536                          ^O20000,CMP2
006360                                          BEQ     OUTO            ; IF = TO, BRANCH
006360  010000  006364                          ^O10000,OUTO
006362                                          BCC     FNDLOP          ; IF LBN > ENDING LBN, BRANCH
006362  040000  006353                          ^O40000,FNDLOP
006364                                  OUTO:   RETURN                  ; RETURN TO CALLING PROGRAM
006364  000000  000000                          ^O0,0
```

```
                                        .SBTTL  MAXMUM - FIND HOW MANY SECTORS TO READ/WRITE
006366                                  MAXMUM:
                                        ;
                                        ;       MAXMUM WILL WRITE THE MAXIMUM NUMBER OF SECTORS IF IN INITIAL WRITE,
                                        ;       OTHERWISE IT WILL R/W ONE TRACK
                                        ;
006366  115000  003006                  TST     M.PARM          ; SEE IF INITIAL WRITE IS IN PROGRESS
006370                                  BMI     1$              ; IF SO, BRANCH
006370  070000  006410                  ^070000,1$
006372                                  ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
006372  115000  003002                  TST     SCR1            ; SEE IF FIRST TIME ON THIS UNIT
006374                                  BEQ     2$              ; IF NOT, BRANCH
006374  010000  006402                  ^010000,2$
006376                                  CALL    NXTTRK          ; ALIGN WITH NEXT TRACK
006376  020000  006543                  ^020000,NXTTRK
006400                                  BR      3$              ; EXIT
006400  000000  006412                  ^00,3$
006402  106647  000006          2$:     CMP     S.TRKL(R4),R0   ; SEE IF TRACK GREATER THAN MAX
006404                                  BPL     1$              ; IF LESS THAN TRACK LENGTH, BRANCH
006404  030000  006410                  ^030000,1$
006406  104647  000006                  MOV     S.TRKL(R4),R0   ; R/W ONE TRACK
006410  100657  000023          1$:     MOV     R0,U.TSEC(R5)   ; SAVE NUMBER OF SECTORS TO R/W
006412                          3$:     RETURN                  ; RETURN TO CALLING PROGRAM
006412  000000  000000                  ^00,0
```

```
                                        .SBTTL  PREVBE - MOVE TO PREVIOUS BEGIN/END SET
006414                                  PREVBE:
                                        ;
                                        ;       PREVBE IS CALLED IF THE LBN'S ARE DECREASING, AND THE LAST LBN
                                        ;       WAS THE FIRST LBN WITHIN A B/E SET.  PREVBE FINDS THE PREVIOUS
                                        ;       B/E SET, OR, IF IT IS THE LAST ON THE SUBUNIT, FINDS THE LAST B/E
                                        ;       SET OF THE PREVIOUS UNIT.  IF IT IS THE FIRST BE SET OF THE FIRST
                                        ;       UNIT, THE DIRECTION OF THE LBN'S ARE REVERSED (THEY BEGIN GOING UP
                                        ;       THE UNIT)
                                        ;
006414  115003                          TST     R3              ; SEE IF ALLREADY ON 1ST B/E SET
006415                                  BNE     5$              ; IF NOT, BRANCH
006415  050000  006427                  ^050000,5$
006417  104657  000046                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
006421  101207  004000                  BIS     #NEWSUB,R0      ; MARK AS GOING ON TO THE NEXT UNIT
006423  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE
006425                                  BR      6$              ; EXIT
006425  000000  006456                  ^00,6$
006427  107201  000006          5$:     SUB     #6,R1           ; POINT TO PREVIOUS ENDING SET
006431                                  CALL    MOV2            ; MOVE ENDING SET TO LBN
006431  020000  006527                  ^020000,MOV2
006433  104217                          MOV     (R1)+,R0        ; R0 HAS LO ENDING LBN
006434  104212                          MOV     (R1)+,R2        ; R2 HAS HI ENDING LBN
006435  107217                          SUB     (R1)+,R0        ; SUBTRACT LO BEGINNING LBN FROM LO ENDING
006436                                  BMI     4$              ; IF RESULT IS NEGATIVE, BRANCH
006436  070000  006451                  ^070000,4$
006440                                  BCC     1$              ; IF NO CARRY, BRANCH
006440  040000  006443                  ^040000,1$
006442  117402                          DEC     R2              ; PROPOGATE CARRY
006443  104111                  1$:     MOV     (R1),R1         ; GET BEGINNING BN
006444  103201  170000                  BIC     #^CHBHINB,R1    ; CLEAR EOL FLAG, IF ANY
006446  107012                          SUB     R1,R2           ; SUBTRACT HI ORDER BN FROM HI BN
006447                                  BEQ     2$              ; IF ZERO, BRANCH
006447  010000  006453                  ^010000,2$
006451  104207  077776          4$:     MOV     #77776,R0       ; MOVE MAXIMUM COUNT TO R0
006453  115407                  2$:     INC     R0              ; TO MAKE IT AN INCLUSIVE SUBTRACT
006454                          3$:     CALL    NXTTRK          ; SET MAXIMUM COUNT (U.TSEC AND U.MSEC)
006454  020000  006543                  ^020000,NXTTRK
006456                          6$:     RETURN                  ; RETURN TO CALLING PROGRAM
006456  000000  000000                  ^00,0
```

NEXTBE - MOVE TO NEXT BEGIN/END SET

```
                                        .SBTTL  NEXTBE - MOVE TO NEXT BEGIN/END SET
006460                                  NEXTBE:
                                        ;
                                        ;       NEXTBE IS CALLED WHEN THE CURRENT LBN IS THE LAST OF ITS B/E SET.
                                        ;       NEXTBE GOES TO THE NEXT B/E SET FOR THE NEXT LBN.  IF THE LBN IS
                                        ;       THE LAST ON THE SUBUNIT, IT GOES TO THE FIRST B/E SET OF THE NEXT
                                        ;       UNIT.  IF IT IS THE LAST B/E SET ON THE LAST UNIT, THE DIRECTION IS
                                        ;       CHANGED (DOWN), AND THE INITIAL WRITE BIT IS CLEARED IN CASE AN
                                        ;       INITIAL WRITE IS IN  PROGRESS
                                        ;
006460  104617  000003                          MOV     3(R1),R0        ; SEE IF THIS IS LAST B/E SET ON THIS UNIT
006462                                          BPL     1$              ; IF SO, BRANCH
006462  030000  006474                          ^030000,1$
006464  104657  000046                          MOV     U.PARM(R5),R0   ; GET SUBUNIT PARAMETERS
006466  101207  004000                          BIS     #NEWSUB,R0      ; MARK AS GOING ON TO A NEW SUBUNIT
006470  100657  000046                          MOV     R0,U.PARM(R5)   ; SAVE
006472                                          BR      7$              ; EXIT
006472  000000  006525                          ^00,7$
006474  105201  000006              1$:         ADD     #6,R1           ; POINT TO NEXT BEGINNING LBN
006476                                          CALL    MOV2            ; MOVE BEGINNING LBN TO LBN
006476  020000  006527                          ^020000,MOV2
006500  107201  000002                          SUB     #2,R1           ; R1 POINTS TO END SET
006502  104217                                  MOV     (R1)+,R0        ; R0 HAS LO ENDING LBN
006503  104213                                  MOV     (R1)+,R3        ; R3 HAS HI ENDING SET
006504  107217                                  SUB     (R1)+,R0        ; SUBTRACT BEGINNING LBN
006505                                          BMI     5$              ; IF RESULT IS NEGATIVE, BRANCH
006505  070000  006520                          ^070000,5$
006507                                          BCC     2$              ; IF NO CARRY, BRANCH
006507  040000  006512                          ^040000,2$
006511  117403                                  DEC     R3              ; PROPOGATE CARRY
006512  ;04111                      2$:         MOV     (R1),R1         ; GET BEGINNING BN
006513  103201  170000                          BIC     #^CHBHINB,R1    ; CLEAR EOL FLAG, IF ANY
006515  107013                                  SUB     R1,R3           ; SUBTRACT HI ORDER BN FROM HI BN
006516                                          BEQ     3$              ; IF ZERO, BRANCH
006516  010000  006522                          ^010000,3$
006520  104207  077776              5$:         MOV     #77776,R0       ; MOVE MAXIMUM COUNT TO R0
006522  115407                      3$:         INC     R0              ; INCREMENT R0 (INCLUSIVE SUBTRACT)
006523                              4$:         CALL    NXTTRK          ; SET MAXIMUM COUNT (U.TSEC AND U.MSEC)
006523  020000  006543                          ^020000,NXTTRK
006525                              7$:         RETURN                  ; RETURN TO CALLING PROGRAM
006525  000000  000000                          ^00,0
```

```
                                .SBTTL  MOV2    - 28 BIT MOVE
006527                          MOV2:
                                ;
                                ;       MOV2 COPIES A 28 BIT NUMBER POINTED TO BY R1 TO TWO WORDS POINTED
                                ;       TO BY R2.  BITS 31-28 ARE CLEARED BEFORE COPYING
                                ;
006527                          PUSH    R0              ; SAVE R0
006527  100467                                                  MOV R0,-(SP)
006530  104117                  MOV     (R1),R0         ; MOVE SOURCE LOW ORDER WORD TO R0
006531  100127                  MOV     R0,(R2)         ; MOVE R0 TO DESTINATION LOW ORDER WORD
006532  104617  000001          MOV     1(R1),R0        ; MOVE SOURCE HIGH ORDER WORD TO R0
006534  103207  170000          BIC     #^CHBHINB,R0    ; STRIP OFF UNUSED BITS
006536  100627  000001          MOV     R0,1(R2)        ; MOVE R0 TO DESTINATION HIGH ORDER WORD
006540                          POP     R0              ; RESTORE R0
006540  104267                                                  MOV (SP)+,R0
006541                          RETURN                  ; RETURN TO CALLING PROGRAM
006541  000000  000000          ^O00,0
```

```
                                .SBTTL  NXTTRK - FIND HOW MANY SECTORS TO R/W TO ALIGN WITH NEXT TRACK BOUNDRY
006543                          NXTTRK:
                                ;
                                ;       COMPUTE HOW MANY SECTORS TO READ/WRITE TO GET ON A TRACK BOUNDRY
                                ;       THIS WILL ALLOW SEQUENTIAL READS AND WRITES TO BE PERFORMED
                                ;       WITHOUT ANY 'BACKWARDS' SEEKING (EXCEPT IN THE CASE OF REVECTORS)
                                ;
006543                                  PUSH    R0              ; SAVE R0
006543  100467                                                                  MOV R0,-(SP)
006544  115000  003006                  TST     M.PARM          ; SEE IF INITIAL WRITE
006546                                  BMI     7$              ; IF SO, EXIT (WRITE MAXIMUM POSSIBLE)
006546  070000  006703                  ^070000,7$
006550                                  ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
006550  104651  000051                  MOV     U.MBN(R5),R1    ; GET LO ORDER SECTOR TO R/W
006552  104010  002760                  MOV     R1,CURBN        ; MOVE TO CALC AREA
006554  104651  000052                  MOV     U.MBN+1(R5),R1  ; GET HI ORDER SECTOR TO R/W
006556  104010  002761                  MOV     R1,CURBN+1      ; MOVE TO CALC AREA
006560  114007                          CLR     R0              ; TELL CALC TO SETUP ALL REQUIRED PARAMETERS
006561                                  CALL    CALC            ; CALC TRACK THAT SECTOR IS ON
006561  020000  002604                  ^020000,CALC
006563  104207  006733                  MOV     #CYLSCR,R0      ; R0 POINTS TO SCRATCH SPACE
006565  104201  002764                  MOV     #CYL,R1         ; R1 POINTS TO CYL, GRP AND TRACK
006567  104202  000004                  MOV     #4,R2           ; MOVE 4 WORDS
006571  104213                  8$:     MOV     (R1)+,R3        ; GET WORD
006572  100273                          MOV     R3,(R0)+        ; SAVE
006573  117402                          DEC     R2              ; DECREMENT COUNT
006574                                  BNE     8$              ; IF INCOMPLETE, BRANCH
006574  050000  006571                  ^050000,8$
006576  104207  000020                  MOV     #20,R0          ; START INCREMENT BY 10 (16 DECIMAL)
006600  104651  000046          1$:     MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
006602                                  CALL    ID              ; INCREMENT OR DECREMENT (DEPENDING ON DIRECTION)
006602  020000  006707                  ^020000,ID
006604                                  PUSH    R0              ; SAVE R0
006604  100467                                                                  MOV R0,-(SP)
006605                                  CALL    CALC            ; CALCULATE NEW CYL/GRP/TRK
006605  020000  002604                  ^020000,CALC
006607  104207  006733                  MOV     #CYLSCR,R0      ; R0 POINTS TO SCRATCH SPACE
006611  104201  002764                  MOV     #CYL,R1         ; R1 POINTS TO CYL, GRP AND TRACK
006613  104202  000004                  MOV     #4,R2           ; MOVE 4 WORDS
006615  104213                  9$:     MOV     (R1)+,R3        ; GET WORD
006616  106273                          CMP     (R0)+,R3        ; COMPARE
006617                                  BNE     2$              ; IF CHANGE, BRANCH
006617  050000  006633                  ^050000,2$
006621  117402                          DEC     R2              ; DECREMENT COUNT
006622                                  BNE     9$              ; IF INCOMPLETE, BRANCH
006622  050000  006615                  ^050000,9$
006624                                  POP     R0              ; RESTORE STEPSIZE
006624  104267                                                                  MOV (SP)+,R0
006625  106207  000020                  CMP     #20,R0          ; SEE IF ORIGINAL STEPSIZE
006627                                  BEQ     1$              ; IF SO, BRANCH
006627  010000  006600                  ^010000,1$
006631                                  BR      3$              ; BRANCH
006631  000000  006646                  ^00,3$
006633                          2$:     POP     R0              ; RESTORE STEPSIZE
006633  104267                                                                  MOV (SP)+,R0
006634  106207  000001                  CMP     #1,R0           ; SEE IF LAST STEPSIZE
006636                                  BEQ     4$              ; FOUND FIRST SECTOR ON ANOTHER TRACK
```

```
                006636  010000  006656                  ^010000,4$
     45 006640  104201  177777          MOV     #-1,R1          ; SET UP FOR COMPLEMENT
     46 006642  103651  000046          BIC     U.PARM(R5),R1   ; COMPLEMENT
     47 006644                          CALL    ID
                006644  020000  006707  ^020000,ID
     48 006646  110607          5$:     ROR     R0              ; ROTATE TO HALVE STEPSIZE
     49 006647  103207  177760          BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
     50 006651                          BNE     1$              ; IF NON-ZERO, LOOP
                006651  050000  006600  ^050000,1$
     51 006653  115407                  INC     R0              ; MAKE R3 1
     52 006654                          BR      1$              ; LOOP
                006654  000000  006600  ^00,1$
     53 006656  104653  000046  4$:     MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
     54 006660  102203  010000          BIT     #DIREC,R3       ; FIND WHAT DIRECTION YOU'RE GOING IN
     55 006662                          BEQ     5$              ; IF UP, BRANCH
                006662  010000  006672  ^010000,5$
     56 006664  104653  000051          MOV     U.MBN(R5),R3    ; GET LO ORDER STARTING BN
     57 006666  107303  002760          SUB     CURBN,R3        ; SUBTRACT ENDING
     58 006670                          BR      6$              ; GO WITH COMPUTED NUMBER
                006670  000000  006676  ^00,6$
     59 006672  104303  002760  5$:     MOV     CURBN,R3        ; GET LO ORDER ENDING BN
     60 006674  107653  000051          SUB     U.MBN(R5),R3    ; SUBTRACT STARTING BN
     61 006676                  6$:     POP     R0              ; GET MAX SECTORS THAT CAN BE R/W
                006676  104267                                                      MOV (SP)+,R0
     62 006677  106073                  CMP     R0,R3           ; COMPARE
     63 006700                          BMI     7$              ; IF CAN WRITE MORE THAN POSSIBLE, BRANCH
                006700  070000  006703  ^070000,7$
     64 006702  104037                  MOV     R3,R0           ; SETUP TO WRITE TO EOT
     65 006703  100657  000023  7$:     MOV     R0,U.TSEC(R5)   ; SAVE IN SECTORS TO R/W
     66 006705                          RETURN                  ; RETURN TO CALLING PROGRAM
                006705  000000  000000  ^00,0
     67
     68
     69 006707                  ID:
     70 006707  102201  010000          BIT     #DIREC,R1       ; SEE IF GOING UP OR DOWN
     71 006711                          BEQ     1$              ; IF UP, BRANCH
                006711  010000  006723  ^010000,1$
     72 006713  107070  002760          SUB     R0,CURBN        ; DECREMENT BN
     73 006715                          BCC     2$              ; IF NO BORROW, EXIT
                006715  040000  006731  ^040000,2$
     74 006717  117400  002761          DEC     CURBN+1         ; PROPOGATE BORROW
     75 006721                          BR      2$              ; EXIT
                006721  000000  006731  ^00,2$
     76 006723  105070  002760  1$:     ADD     R0,CURBN        ; INCREMENT BN
     77 006725                          BCC     2$              ; IF NO CARRY, BRANCH
                006725  040000  006731  ^040000,2$
     78 006727  115400  002761          INC     CURBN+1         ; PROPOGATE CARRY
     79 006731                  2$:     RETURN
                006731  000000  000000  ^00,0
     80
     81
     82 006733  000000  CYLSCR: .WORD   0               ; SCRATCH AREA FOR STORING CYL, GRP AND TRK
     83 006734  000000          .WORD   0
     84 006735  000000          .WORD   0
     85 006736  000000          .WORD   0
```

```
                                        .SBTTL  ********** OVERLAY MODULE RNDTG - RANDOMLY GET NEXT TRACK/GROUP SECTOR
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;
                                        ;
                                        ;
 006737                                 RNDTG:
                                        ;
                                        ;       COME UP WITH A RANDOM BN TO TEST USING THE TRACK/GROUP SETS
                                        ;
                                        ;       FIRST COME UP WITH A RANDOM COUNT <= MAXIMUM COUNT
                                        ;
                                        .ENABL  LSB
 006737                                 1$:     CALL    RANDOM          ; GET RANDOM NUMBER
 006737  020000  005563                         ^O020000,RANDOM
 006741  114002                                 CLR     R2              ; HI ORDE WORD IS ZERO
 006742  104301  003007                         MOV     LOSEED,R1       ; LOSEED IN R1
 006744  104647  000014                         MOV     S.MCNT+1(R4),R0 ; GET MAXIMUM HI ORDER WORD
 006746                                         BEQ     3$              ; IF ZERO, BRANCH
 006746  010000  006765                         ^O010000,3$
 006750                                         CALL    MASK            ; CREATE MASK FOR RANDOM NUMBER
 006750  020000  005630                         ^O020000,MASK
 006752  104302  003010                         MOV     HISEED,R2       ; PUT NUMBER IN R2
 006754  103072                                 BIC     R0,R2           ; STRIP OFF UNUSED BITS
 006755  106642  000014                         CMP     S.MCNT+1(R4),R2 ; SEE IF MAX EXCEEDED
 006757                                         BCS     1$              ; IF SO, TRY AGAIN
 006757  040000  006763                         ^O040000,..+3
 006761  000000  006737                         ^O0,1$
 006763                                         BNE     5$              ; IF NOT EQUAL, EXIT
 006763  050000  007000                         ^O050000,5$
 006765  104647  000013                 3$:     MOV     S.MCNT(R4),R0   ; GET LO ORDER MAX
 006767                                         CALL    MASK            ; GET RANDOM MASK
 006767  020000  005630                         ^O020000,MASK
 006771  103071                                 BIC     R0,R1           ; STRIP OFF BITS
 006772  106641  000013                         CMP     S.MCNT(R4),R1   ; CHECK WITH MAX
 006774                                         BCS     1$              ; IF GREATER THAN MAX, DO AGAIN
 006774  040000  007000                         ^O040000,..+3
 006776  000000  006737                         ^O0,1$
```

```
                                        ;
                                        ;       NOW FIND THE BN ASSOCIATED WITH THAT COUNT
                                        ;
007000                          5$:     PUSH    <R5,R4>         ; PUSH THE UNIT AND SUBUNIT POINTERS
007000  100465                                                                  MOV R5,-(SP)
007001  100464                                                                  MOV R4,-(SP)
007002  104205  000015                  MOV     #S.TGOF,R5      ; R5 WILL POINT TO THE TRACK/GROUP OFFSETS
007004  105045                          ADD     R4,R5           ; R5 POINTS TO THE TRACK/GROUP ORIGINAL OFFSET
007005  104257                          MOV     (R5)+,R0        ; GET LO ORDER OFFSET, MOVE TO RUNNING TOTAL
007006  104253                          MOV     (R5)+,R3        ; GET HI ORDER OFFSET, MOVE TO RUNNING TOTAL
007007  107201  000001          6$:     SUB     #1,R1           ; DECREMENT LO COUNT BY ONE
007011                                  BCC     7$              ; IF NO BORROW, BRANCH
007011  040000  007021                  ^040000,7$
007013  107202  000001                  SUB     #1,R2           ; DECREMENT HI COUNT BY ONE
007015                                  BCS     9$              ; IF COUNT EXHAUSTED, BRANCH
007015  040000  007021                  ^040000,..+3
007017  000000  007036                  ^00,9$
007021  104254                  7$:     MOV     (R5)+,R4        ; GET T/G OFFSET
007022                                  BNE     8$              ; IF NOT END OF LIST, BRANCH
007022  050000  007030                  ^050000,8$
007024  104205  000017                  MOV     #S.TGSS,R5      ; R5 WILL POINT TO START OF T/G LIST
007026  105165                          ADD     (SP),R5         ; R5 POINTS TO START OF T/G LIST
007027  104254                          MOV     (R5)+,R4        ; GET T/G OFFSET
007030  105047                  8$:     ADD     R4,R0           ; ADD TO RUNNING TOTAL
007031                                  BCC     6$              ; IF NO CARRY, BRANCH
007031  040000  007007                  ^040000,6$
007033  115403                          INC     R3              ; PROPOGATE CARRY
007034                                  BR      6$              ; BRANCH
007034  000000  007007                  ^00,6$
007036                          9$:     POP     <R4,R5>         ; RESTORE
007036  104264                                                                  MOV (SP)+,R4
007037  104265                                                                  MOV (SP)+,R5
```

```
                                        ;
                                        ;       NOW FIND A RANDOM OFFSET INTO THE RANDOM TRACK OR GROUP
                                        ;
007040                                  PUSH    R0              ; STORE LO ORDER TOTAL ON STACK
007040  100467                                                                  MOV R0,-(SP)
007041  104642  000006                  MOV     S.TRKL(R4),R2   ; R2 HAS TRACK LENGTH
007043  104147                          MOV     (R4),R0         ; GET SUBUNIT PARAMETERS
007044                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
007044  102207  000020                  BIT     #TRACKS,R0      ; SEE IF USING TRACKS
007046                                  BNE     11$             ; IF SO, BRANCH
007046  050000  007064                  ^050000,11$
007050  104641  000007                  MOV     S.SCHR(R4),R1   ; R1 POINTS TO SUBUNIT PARAMETERS
007052  104611  000003                  MOV     TRKGRP(R1),R1   ; R1 HAS NUMBER OF TRACKS/GROUP
007054  103201  177400                  BIC     #HIBYTE,R1      ; CLEAR UNUSED BITS
007056  114007                          CLR     R0              ; CLEAR RUNNING TOTAL
007057  105027                  10$:    ADD     R2,R0           ; FIND SECTORS/GROUP
007060  117401                          DEC     R1              ; DECREMENT COUNT
007061                                  BNE     10$             ; IF UNEXHAUSTED, BRANCH
007061  050000  007057                  ^050000,10$
007063  104072                          MOV     R0,R2           ; MOVE COUNT TO R2 (MATCH WITH TRACKS ABOVE)
007064                          11$:    PUSH    R2              ; SAVE ON STACK
007064  100462                                                                  MOV R2,-(SP)
007065  117402                          DEC     R2              ; ADJUST COUNT
007066  104027                          MOV     R2,R0           ; MOVE TO MASKING REGISTER
007067                                  CALL    MASK            ; FIND MASK
007067  020000  005630                  ^020000,MASK
007071                          12$:    CALL    RANDOM          ; GET RANDOM NUMBER
007071  020000  005563                  ^020000,RANDOM
007073  103071                          BIC     R0,R1           ; STRIP OFF UNUSED BITS
007074  106021                          CMP     R2,R1           ; SEE IF RANDOM NUMBER SMALL ENOUGH
007075                                  BCS     12$             ; IF NOT, BRANCH
007075  040000  007101                  ^040000,.+3
007077  000000  007071                  ^O0,12$
007101                                  POP     R2              ; GET NUMBER OF SECTORS
007101  104262                                                                  MOV (SP)+,R2
007102  107012                          SUB     R1,R2           ; R2 IS NOW SECTORS REMAINING IN T/G
007103  105261                                                                  ADD     (SP)+,R1
007104  100651  000051                  MOV     R1,U.MBN(R5)    ; SAVE
007106                                  BCC     13$             ; IF NO CARRY, BRANCH
007106  040000  007111                  ^040000,13$
007110  115403                          INC     R3              ; PROPOGATE CARRY
007111  100653  000052          13$:    MOV     R3,U.MBN+1(R5)  ; SAVE HI ORDER STARTING BN
```

```
                                ;
                                ;       NOW COME UP WITH A RANDOM NUMBER OF SECTORS TO R/W WITHOUT GOING
                                ;       OFF THE TRACK/GROUP THAT WE'RE ON
                                ;
007113  106202  000016                  CMP     #16.,R2         ; ONLY WRITE A MAXIMUM OF 16 SECTORS
007115                                  BPL     15$             ; IF LESS THAN 16, BRANCH
007115  030000  007123                  ^O030000,15$
007117  104207  177740                  MOV     #MAXMSK,R0      ; SET READ/WRITE LIMIT
007121                                  BR      14$             ; BRANCH
007121  000000  007126                  ^O00,14$
007123  104027                  15$:    MOV     R2,R0           ; R0 IS SECTORS LEFT ON THIS T/G
007124                                  CALL    MASK            ; GET MASK
007124  020000  005630                  ^O020000,MASK
007126                          14$:    CALL    RANDOM          ; GET RANDOM NUMBER
007126  020000  005563                  ^O020000,RANDOM
007130  103071                          BIC     R0,R1           ; STRIP OFF UNUSED BITS
007131  115401                          INC     R1              ; MUST BE GREATER THAN ZERO
007132  106021                          CMP     R2,R1           ; SEE IF SMALL ENOUGH
007133                                  BCS     14$             ; IF NOT, BRANCH
007133  040000  007137                  ^O040000,.+3
007135  000000  007126                  ^O00,14$
007137  100651  000023                  MOV     R1,U.TSEC(R5)   ; SAVE SECTORS TO WRITE
007141  104207  005403                  MOV     #AFTOP,R0       ; NEXT MODULE IS AFTOP
007143  114001                          CLR     R1              ; IMMIDATE CALL
007144  114002                          CLR     R2              ; NO ERRORS
007145                                  BR      JMPRET          ; RETURN TO SEQNCR
007145  000000  003746                  ^O00,JMPRET
                                .DSABL  LSB
```

```
                                        .SBTTL  ********** OVERLAY MODULE SEQTG - GET NEXT SEQUENTIAL TRACK/GROUP SECTOR
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;
                                        ;
007147                                  SEQTG:
                                        ;
                                        ;       FIND THE NEXT TRACK/GROUP TO TEST SEQUENTIALLY
                                        ;
                                                .ENABL  LSB
007147  104657  000046                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
007151  104141                          MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
007152                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
007152  102207  010000                  BIT     #DIREC,R0       ; SEE IF GOING UP (INITIAL WRITE) OR DOWN
007154                                  BEQ     3$              ; IF GOING UP OR INITIAL WRITE, BRANCH
007154  010000  007172                  ^010000,3$
007156  102201  000020                  BIT     #TRACKS,R1      ; SEE IF TRACKS OR GROUPS
007160                                  BEQ     2$              ; IF GROUPS, BRANCH
007160  010000  007166                  ^010000,2$
007162                                  CALL    DOWNT           ; GO DOWN A TRACK
007162  020000  007333                  ^020000,DOWNT
007164                                  BR      5$              ; CALCULATE NUMBER OF SECTORS TO READ/WRITE
007164  000000  007204                  ^00,5$
007166                          2$:     CALL    DOWNG           ; GO DOWN A GROUP
007166  02C000  007525                  ^020000,DOWNG
007170                                  BR      5$              ; CALCULATE NUMBER OF SECTORS TO READ/WRITE
007170  000000  007204                  ^00,5$
007172  102201  000020          3$:     BIT     #TRACKS,R1      ; SEE IF TRACKS OR GROUPS
007174                                  BEQ     4$              ; IF GROUPS, BRANCH
007174  010000  007202                  ^010000,4$
007176                                  CALL    UPT             ; GO UP A TRACK
007176  020000  007243                  ^020000,UPT
007200                                  BR      5$              ; CALCULATE NUMBER OF SECTORS TO READ/WRITE
007200  000000  007204                  ^00,5$
007202                          4$:     CALL    UPG             ; GO UP A GROUP
007202  020000  007427                  ^020000,UPG
007204  104642  000006          5$:     MOV     S.TRKL(R4),R2   ; GET SECTORS/TRACK
007206  104141                          MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
007207                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
007207  115000  003006                  TST     M.PARM          ; SEE IF INITIAL WRITE IN PROGRESS
007211                                  BPL     7$              ; IF NOT, BRANCH
007211  030000  007233                  ^030000,7$
007213                                  ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
007213  102201  000020                  BIT     #TRACKS,R1      ; SEE IF TRACKS IN USE
007215                                  BNE     7$              ; IF SO, BRANCH
007215  050000  007233                  ^050000,7$
007217  104027                          MOV     R2,R0           ; COPY SECTORS/TRACK TO R0
007220  104641  000007                  MOV     S.SCHR(R4),R1   ; GET POINTER TO SUBUNIT CHARACTERISTICS
007222  104611  000003                  MOV     TRKGRP(R1),R1   ; GET TRACKS/GROUP
007224  103201  177400                  BIC     #HIBYTE,R1      ; CLEAR UNUSED BITS
007226  114002                          CLR     R2              ; CLEAR RUNNING TOTAL
007227  105072                  6$:     ADD     R0,R2           ; ADD TO RUNNING TOTAL
007230  117401                          DEC     R1              ; DECREMENT COUNT
007231                                  BNE     6$              ; IF .NCOMPLETE, BRANCH
```

```
            007231  050000  007227                  ^050000,6$
         64 007233  100652  000023          7$:     MOV     R2,U.TSEC(R5)   ; SAVE NUMBER OF SECTORS TO PROCESS
         65 007235  104207  005403          8$:     MOV     #AFTOP,R0       ; NEXT MODULE IS AFTOP
         66 007237  114001                          CLR     R1              ; IMMIDATE CALL
         67 007240  114002                          CLR     R2              ; NO ERRORS
         68 007241                                  BR      JMPRET          ; RETURN TO SEQNCR
            007241  000000  003746                  ^O0,JMPRET
         69                                         .DSABL  LSB
```

```
                                .SBTTL  UPT     - MOVE UP ONE TRACK
007243                          UPT:
                                ;
                                ;       POINT U.MBN TO THE NEXT TRACK TO TEST (IT DOESN'T HAVE TO BE A
                                ;       TRACK, BUT IF GROUPS, U.MBN MUST BE ADJUSTED BACK TO THE FIRST
                                ;       BN IN THE GROUP
                                ;
007243  115000  003002                  TST     SCR1            ; SEE IF NEW SUBUNIT
007245                                  BEQ     10$             ; IF NOT, BRANCH
007245  010000  007253                  ^O10000,10$
007247                                  CALL    NEWUPT          ; SETUP NEW SUBUNIT
007247  020000  007573                  ^O20000,NEWUPT
007251                                  BR      4$              ; EXIT
007251  000000  007331                  ^O00,4$
007253  104657  000015          10$:    MOV     U.CCNT(R5),R0   ; GET LO ORDER COUNT
007255  107207  000001                  SUB     #1,R0           ; DECREMENT
007257  100657  000015                  MOV     R0,U.CCNT(R5)   ; SAVE
007261                                  BCC     2$              ; IF NO BORROW, BRANCH
007261  040000  007303                  ^O40000,2$
007263  104657  000016                  MOV     U.CCNT+1(R5),R0 ; GET HI ORDER COUNT
007265  107207  000001                  SUB     #1,R0           ; PROPOGATE BORROW
007267                                  BCC     1$              ; IF NO BORROW, BRANCH
007267  040000  007301                  ^O40000,1$
007271  104657  000046                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
007273  101207  004000                  BIS     #NEWSUB,R0      ; FLAG AS TO GO ONTO NEXT SUBUNIT
007275  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE
007277                                  BR      4$              ; EXIT
007277  000000  007331                  ^O00,4$
007301  100657  000016          1$:     MOV     R0,U.CCNT+1(R5) ; SAVE
007303  104657  000017          2$:     MOV     U.PCTG(R5),R0   ; GET POINTER TO CURRENT TRACK/GROUP
007305  104271                          MOV     (R0)+,R1        ; GET OFFSET TO NEXT TRACK
007306                                  BNE     3$              ; IF OFFSET, BRANCH
007306  050000  007314                  ^O50000,3$
007310  104207  000017                  MOV     #S.TGSS,R0      ; R0 WILL POINT TO START OF OFFSETS
007312  105047                          ADD     R4,R0           ; R0 POINTS TO OFFSETS
007313  104271                          MOV     (R0)+,R1        ; R1 HAS OFFSET
007314  100657  000017          3$:     MOV     R0,U.PCTG(R5)   ; SAVE POINTER TO CURRENT TRACK/GROUP
007316  105651  000051                  ADD     U.MBN(R5),R1    ; ADD OFFSET TO CURRENT BN
007320  100651  000051                  MOV     R1,U.MBN(R5)    ; SAVE
007322                                  BCC     4$              ; IF NO CARRY, EXIT
007322  040000  007331                  ^O40000,4$
007324  104651  000052                  MOV     U.MBN+1(R5),R1  ; GET HI ORDER BN
007326  115401                          INC     R1              ; PROPOGATE CARRY
007327  100651  000052                  MOV     R1,U.MBN+1(R5)  ; SAVE
007331                          4$:     RETURN                  ; RETURN TO CALLING PROGRAM
007331  000000  000000                  ^O00,0
```

```
                                        .SBTTL  DOWNT  - MOVE DOWN ONE TRACK
007333                                  DOWNT:
                                        ;
                                        ;       MOVE THE MASTER BN DOWN TO TEST THE PROCEEDING TRACK/GROUP
                                        ;
007333  115000  003002                  TST     SCR1            ; SEE IF NEW SUBUNIT
007335                                  BEQ     10$             ; IF NOT, BRANCH
007335  010000  007343                  ^010000,10$
007337                                  CALL    NEWDNT          ; INITIALIZE THIS SUBUNIT
007337  020000  007622                  ^020000,NEWDNT
007341                                  BR      5$              ; EXIT
007341  000000  007425                  ^00,5$
007343  104657  000015          10$:    MOV     U.CCNT(R5),R0   ; GET LO ORDER COUNT
007345  107207  000001                  SUB     #1,R0           ; DECREMENT COUNT
007347  100657  000015                  MOV     R0,U.CCNT(R5)   ; SAVE
007351                                  BCC     2$              ; IF NO BORROW, BRANCH
007351  040000  007373                  ^040000,2$
007353  104657  000016                  MOV     U.CCNT+1(R5),R0 ; GET HI ORDER COUNT
007355  107207  000001                  SUB     #1,R0           ; PROPOGATE BORROW
007357                                  BCC     1$              ; IF NO CARRY, BRANCH
007357  040000  007371                  ^040000,1$
007361  104657  000046                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
007363  101207  004000                  BIS     #NEWSUB,R0      ; FLAG AS TO GO ONTO NEXT SUBUNIT
007365  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE
007367                                  BR      5$              ; EXIT
007367  000000  007425                  ^00,5$
007371  100657  000016          1$:     MOV     R0,U.CCNT+1(R5) ; SAVE
007373  104201  000017          2$:     MOV     #S.TGSS,R1      ; R0 WILL POINT TO START OF T/G OFFSETS
007375  105041                          ADD     R4,R1           ; R0 POINTS TO START OF T/G OFFSETS
007376  104657  000017                  MOV     U.PCTG(R5),R0   ; GET POINTER INTO LIST
007400  106071                          CMP     R0,R1           ; SEE IF AT START OF LIST
007401                                  BNE     4$              ; IF NOT, BRANCH
007401  050000  007407                  ^050000,4$
007403  115407          3$:             INC     R0              ; R0 POINTS TO NEXT WORD
007404  104171                          MOV     (R0),R1         ; SEE IF END OF LIST
007405                                  BNE     3$              ; IF NOT, BRANCH
007405  050000  007403                  ^050000,3$
007407  104651  000051          4$:     MOV     U.MBN(R5),R1    ; GET LO ORDER MASTER BN
007411  107471                          SUB     -(R0),R1        ; SUBTRACT OFFSET
007412  100651  000051                  MOV     R1,U.MBN(R5)    ; SAVE
007414  100657  000017                  MOV     R0,U.PCTG(R5)   ; SAVE POINTER
007416                                  BCC     5$              ; IF NO BORROW, EXIT
007416  040000  007425                  ^040000,5$
007420  104657  000052                  MOV     U.MBN+1(R5),R0  ; GET HI ORDER WORD
007422  117407                          DEC     R0              ; PROPOGATE BORROW
007423  100657  000052                  MOV     R0,U.MBN+1(R5)  ; SAVE
007425                          5$:     RETURN                  ; RETURN TO CALLING PROGRAM
007425  000000  000000                  ^00,0
```

```
                                        .SBTTL  UPG     - MOVE UP ONE GROUP
007427                                  UPG:
                                        ;
                                        ;       MOVE MASTER BN UP TO THE NEXT GROUP
                                        ;
007427  115000  003002                  10$:    TST     SCR1            ; SEE IF THIS IS A NEW SUBUNIT
007431                                          BEQ     10$             ; IF NOT, BRANCH
007431  010000  007441                          ^010000,10$
007433                                          CALL    NEWUPT          ; INITILIZE THE SUBUNIT FOR TRACKS
007433  020000  007573                          ^020000,NEWUPT
007435                                          CALL    SETSEC          ; INITILIZE THE SUBUNIT FOR GROUPS
007435  020000  007705                          ^020000,SETSEC
007437                                          BR      5$              ; EXIT
007437  000000  007523                          ^00,5$
007441  102201  040000                  10$:    BIT     #INITW,R1       ; SEE IF AN INITIAL WRITE IS BEING DONE
007443                                          BNE     4$              ; IF SO, BRANCH
007443  050000  007520                          ^050000,4$
007445  104641  000006                          MOV     S.TRKL(R4),R1   ; R1 HAS TRACK LENGTH
007447  104657  000020                          MOV     U.CTRK(R5),R0   ; GET TRACK COUNT
007451  117407                                  DEC     R0              ; DECREMENT COUNT
007452                                          BEQ     1$              ; IF COUNT EXHAUSTED, BRANCH
007452  010000  007473                          ^010000,1$
007454  100657  000020                          MOV     R0,U.CTRK(R5)   ; SAVE
007456  105651  000051                          ADD     U.MBN(R5),R1    ; MOVE BN TO NEXT TRACK
007460  100651  000051                          MOV     R1,U.MBN(R5)    ; SAVE
007462                                          BCC     5$              ; EXIT
007462  040000  007523                          ^040000,5$
007464  104657  000052                          MOV     U.MBN+1(R5),R0  ; GET HI ORDER BN
007466  115407                                  INC     R0              ; PROPOGATE CARRY
007467  100657  000052                          MOV     R0,U.MBN+1(R5)  ; SAVE
007471                                          BR      5$              ; EXIT
007471  000000  007523                          ^00,5$
007473                                  1$:     CALL    SETSEC          ; SETUP NEXT TRACK/GROUP COUNT
007473  020000  007705                          ^020000,SETSEC
007475  114002                                  CLR     R2              ; CLEAR RUNNING TOTAL
007476  117407                          2$:     DEC     R0              ; DECREMENT COUNT
007477                                          BEQ     3$              ; IF COUNT EXPIRED, BRANCH
007477  010000  007504                          ^010000,3$
007501  105012                                  ADD     R1,R2           ; ADD SECTORS/TRACK TO RUNNING TOTAL
007502                                          BR      2$              ; LOOP
007502  000000  007476                          ^00,2$
007504  104657  000051                  3$:     MOV     U.MBN(R5),R0    ; GET LO ORDER MASTER BN
007506  107027                                  SUB     R2,R0           ; ADJUST BACK TO START OF GROUP
007507  100657  000051                          MOV     R0,U.MBN(R5)    ; SAVE
007511                                          BCC     4$              ; IF NO BORROW, BRANCH
007511  040000  007520                          ^040000,4$
007513  104657  000052                          MOV     U.MBN+1(R5),R0  ; GET HI ORDER MASTFR BN
007515  117407                                  DEC     R0              ; PROPOGATE BORROW
007516  100657  000052                          MOV     R0,U.MBN+1(R5)  ; SAVE
007520  114007                          4$:     CLR     R0              ; CLEAR R0 SO UPT DOESN'T DETECT A NEW SUBUNIT
007521                                          CALL    UPT             ; GO TO NEXT GROUP
007521  020000  007243                          ^020000,UPT
007523                                  5$:     RETURN                  ; RETURN TO CALLING PROGRAM
007523  000000  000000                          ^00,0
```

```
                                        .SBTTL  DOWNG  - MOVE DOWN ONE GROUP
007525                                  DOWNG:
                                        ;
                                        ;       MOVE MASTER BN TO THE NEXT LOWER GROUP
                                        ;
007525  115000  003002                          TST     SCR1            ; SEE IF THIS IS A NEW SUBIUNIT
007527                                          BEQ     10$             ; IF NOT, BRANCH
007527  010000  007537                          ^O010000,10$
007531                                          CALL    NEWDNT          ; SETUP THIS SUBUNIT FOR TRACKS
007531  020000  007622                          ^O020000,NEWDNT
007533                                          CALL    LSTTRK          ; SETUP THIS SUBUNIT FOR GROUPS
007533  020000  007717                          ^O020000,LSTTRK
007535                                          BR      4$              ; EXIT
007535  000000  007571                          ^O0,4$
007537  104657  000020                  10$:    MOV     U.CTRK(R5),R0   ; GET TRACK COUNT
007541  117407                                  DEC     R0              ; DECREMENT COUNT
007542                                          BEQ     1$              ; IF EXPIRED, BRANCH
007542  010000  007565                          ^O010000,1$
007544  100657  000020                          MOV     R0,U.CTRK(R5)   ; SAVE
007546  104657  000051                          MOV     U.MBN(R5),R0    ; GET LO ORDER MASTER BN
007550  107647  000006                          SUB     S.TRKL(R4),R0   ; SUBTRACT TRACK LENGTH
007552  100657  000051                          MOV     R0,U.MBN(R5)    ; SAVE
007554                                          BCC     4$              ; EXIT
007554  040000  007571                          ^O040000,4$
007556  104657  000052                          MOV     U.MBN+1(R5),R0  ; GET HI MASTER BN
007560  117407                                  DEC     R0              ; PROPOGATE BORROW
007561  100657  000052                          MOV     R0,U.MBN+1(R5)  ; SAVE
007563                                          BR      4$              ; EXIT
007563  000000  007571                          ^O0,4$
007565                                  1$:     CALL    DOWNT           ; GO DOWN A GROUP
007565  020000  007333                          ^O020000,DOWNT
007567                                          CALL    LSTTRK          ; SET MASTER BN ON LAST TRACK OF GROUP
007567  020000  007717                          ^O020000,LSTTRK
007571                                  4$:     RETURN                  ; RETURN TO CALLING PROGRM
007571  000000  000000                          ^O0,0
```

```
                                        .SBTTL  NEWUPT - INITILIZE PARAMETERS FOR SEQUENTIALLY UP BY TRACKS
007573                                  NEWUPT:
                                        ;
                                        ;       INITIALIZE THIS SUBUNIT FOR SEQUENTIALLY ACCESSING TRACKS IN
                                        ;       ASCENDING ORDER
                                        ;
007573  104647  000013                  MOV     S.MCNT(R4),R0   ; GET MAXIMUM COUNT
007575  100657  000015                  MOV     R0,U.CCNT(R5)   ; SAVE IN CURRENT COUNT
007577  104647  000014                  MOV     S.MCNT+1(R4),R0 ; GET HI ORDER MAXIMUM COUNT
007601  100657  000016                  MOV     R0,U.CCNT+1(R5) ; SAVE IN HI ORDER CURRENT COUNT
007603  104647  000015                  MOV     S.TGOF(R4),R0   ; GET STARTING OFFSET
007605  100657  000051                  MOV     R0,U.MBN(R5)    ; SAVE IN MASTER BN
007607  104647  000016                  MOV     S.TGOF+1(R4),R0 ; GET HI ORDER STARTING OFFSET
007611  100657  000052                  MOV     R0,U.MBN+1(R5)  ; SAVE IN HI ORDER MASTER BN
007613  104207  000017                  MOV     #S.TGSS,R0      ; R0 WILL POINT TO START OF OFFSETS
007615  105047                          ADD     R4,R0           ; R0 POINTS TO START OF OFFSETS
007616  100657  000017                  MOV     R0,U.PCTG(R5)   ; SAVE IN POINTER TO CURRENT TRACK/GROUP
007620                                  RETURN                  ; RETURN TO CALLING PROGRAM
007620  000000  000000                  ^O0,0
```

```
                                .SBTTL  NEWDNT - INITILIZE PARAMETERS FOR SEQUENTIALLY DOWN BY TRACKS
007622                          NEWDNT:
                                ;
                                ;       INITIALIZE THIS SUBUNIT FOR SEQUENTIALLY ACCESSING TRACKS IN
                                ;       DESCENDING ORDER
                                ;
007622                                  PUSH    <R5,R4>         ; SAVE UNIT AND SUBUNIT POINTERS
007622  100465                                                                  MOV R5,-(SP)
007623  100464                                                                  MOV R4,-(SP)
007624  104642  000015                 MOV     S.TGOF(R4),R2   ; R2 HAS LO ORDER STARTING OFFSET
007626  104643  000016                 MOV     S.TGOF+1(R4),R3 ; R3 HAS HI ORDER STARTING OFFSET
007630  104647  000013                 MOV     S.MCNT(R4),R0   ; R0 HAS LO ORDER MAXIMUM COUNT
007632  100657  000015                 MOV     R0,U.CCNT(R5)   ; SAVE LO ORDER MAXIMUM COUNT
007634  104641  000014                 MOV     S.MCNT+1(R4),R1 ; R1 HAS HI ORDER MAXIMUM COUNT
007636  100651  000016                 MOV     R1,U.CCNT+1(R5) ; SAVE HI ORDER MAXIMUM COUNT
007640  105204  000017                 ADD     #S.TGSS,R4      ; R4 POINTS TO START OF OFFSETS
007642  107207  000001         1$:     SUB     #1,R0           ; DECREMENT COUNT
007644                                  BCC     2$              ; IF NO CARRY, BRANCH
007644  040000  007654                 ^O040000,2$
007646  107201  000001                 SUB     #1,R1           ; PROPOGATE BORROW
007650                                  BCS     4$              ; IF COUNT EXHAUSTED, BRANCH
007650  040000  007654                 ^O040000,..+3
007652  000000  007671                 ^O00,4$
007654  104245                  2$:     MOV     (R4)+,R5        ; GET OFFSET
007655                                  BNE     3$              ; IF NOT END-OF-LIST, BRANCH
007655  050000  007663                 ^O050000,3$
007657  104204  000017                 MOV     #S.TGSS,R4      ; R4 WILL POINT TO START OF OFFSETS
007661  105164                          ADD     (SP),R4         ; R4 POINTS TO START OF OFFSETS
007662  104245                          MOV     (R4)+,R5        ; GET OFFSET
007663  105052                  3$:     ADD     R5,R2           ; ADD OFFSET TO BN
007664                                  BCC     1$              ; IF NO CARRY, LOOP
007664  040000  007642                 ^O040000,1$
007666  115403                          INC     R3              ; PROPOGATE CARRY
007667                                  BR      1$              ; LOOP
007667  000000  007642                 ^O00,1$
007671  104665  000001          4$:     MOV     1(SP),R5        ; GET UNIT POINTER
007673  100654  000017                 MOV     R4,U.PCTG(R5)   ; SAVE POINTER TO CURRENT TRACK/GROUP
007675                                  POP     <R4,R5>         ; RESTORE
007675  104264                                                                  MOV (SP)+,R4
007676  104265                                                                  MOV (SP)+,R5
007677  100652  000051                 MOV     R2,U.MBN(R5)    ; SAVE LO ORDER STARTING BN
007701  100653  000052                 MOV     R3,U.MBN+1(R5)  ; SAVE HI ORDER
007703                                  RETURN                  ; RETURN TO CALLING PROGRAM
007703  000000  000000                 ^O00,0
```

```
      1                                    .SBTTL  SETSEC - SETUP TRACK/GROUP COUNT FOR SELECTED GROUPS
      2 007705                             SETSEC:
      3                                    ;
      4                                    ;       MOVE TRACKS/GROUP TO U.CTRK
      5                                    ;
      6 007705  104647  000007                     MOV     S.SCHR(R4),R0   ; R0 POINTS TO SUBUNIT CHARACTERISTICS
      7 007707  104677  000003                     MOV     TRKGRP(R0),R0   ; R0 HAS TRACKS/GROUPS
      8 007711  103207  177400                     BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
      9 007713  100657  000020                     MOV     R0,U.CTRK(R5)   ; SAVE AS TRACK COUNT
     10 007715                                     RETURN                  ; RETURN TO CALLING MODULE
        007715  000000  000000                     ^O0,0
```

```
                                        .SBTTL  LSTTRK - SET MASTER BN TO POIN"  ) LAST TRACK IN THE GROUP
007717                                  LSTTRK:
                                        ;
                                        ;       SET MBN TO LAST TRACK OF GROUP
                                        ;
007717                                          CALL    SETSEC          ; SETUP TRACK/GROUP COUNT
007717  020000  007705                          ^O20000,SETSEC
007721  114002                                 CLR     R2              ; CLEAR RUNNING TOTAL
007722  117407                          2$:     DEC     R0              ; DECREMENT COUNT
007723                                          BEQ     3$              ; IF COUNT EXPIRED, BRANCH
007723  010000  007731                          ^O10000,3$
007725  105642  000006                          ADD     S.TRKL(R4),R2   ; ADD SECTORS/TRACK TO RUNNING LENGTH
007727                                          BR      2$              ; LOOP
007727  000000  007722                          ^O0,2$
007731  105652  000051                  3$:     ADD     U.MBN(R5),R2    ; MBN WILL POINT TO LAST TRACK IN GROUP
007733  100652  000051                          MOV     R2,U.MBN(R5)    ; SAVE
007735                                          BCC     4$              ; IF NO CARRY, EXIT
007735  040000  007742                          ^O40000,4$
007737  104657  000052                          MOV     U.MBN+1(R5),R0  ; GET HI ORDER BN
007741  115407                                  INC     R0              ; PROPOGATE CARRY
007742  100657  000052                  4$:     MOV     R0,U.MBN+1(R5)  ; SAVE
007744                                          RETURN                  ; RETURN TO CALLING MODULE
007744  000000  000000                          ^O0,0
```

```
********** OVERLAY MODULE BUILDP - BUILD THE READ/WRITE LINKED

                                        .SBTTL  ********** OVERLAY MODULE BUILDP - BUILD THE READ/WRITE LINKED LISTS
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;
                                        ;
                                        ;
007746                                  BUILDP:
                                        ;
                                        ;       BUILD THE READ/WRITE LINKED LISTS
                                        ;
                                                .ENABL  LSB
007746                                          CALL    TLKHST          ; COMMUNICATE WITH THE HOST SO NOT TO TIME OUT
007746  020000  001517                          ^O020000,TLKHST
007750  104657  000046                          MOV     U.PARM(R5),R0   ; R0 HAS UNIT PARAMETERS
007752  104650  000022  003020                  MOV     U.MSEC(R5),MAXSEC       ; GET NUMBER OF SECTORS TO READ/WRITE
007755  102207  000200                          BIT     #RBNBN,R0       ; SEE IF RBN TO BE READ/WRITTEN
007757                                          BEQ     1$              ; IF NOT, BRANCH
007757  010000  007771                          ^O010000,1$
007761  104650  000055  002760                  MOV     U.RBN(R5),CURBN ; MOVE LO ORDER TO CURRENT BN
007764  104650  000056  002761                  MOV     U.RBN+1(R5),CURBN+1     ; MOVE HI ORDER TO CURRENT BN
007767                                          BR      4$              ; BRANCH
007767  000000  007777                          ^O0,4$
007771  104650  000053  002760  1$:             MOV     U.CBN(R5),CURBN         ; MOVE LO ORDER TO CURRENT BN
007774  104650  000054  002761                  MOV     U.CBN+1(R5),CURBN+1     ; MOVE LO ORDER TO CURRENT BN
007777  104300  003012  010277  4$:             MOV     MEMPOL,TMEMPL   ; START OF BUILD (INITILIZE MEM POOL)
010002  114007                                  CLR     R0              ; FORCE THE CALCULATION TABLE TO BE FILLED
010003  104070  003015                          MOV     R0,CHAINS       ; MARK CHAIN AS EMPTY
010005                                  5$:     CALL    CALC            ; CALCULATE CYL, TRK AND GRP
010005  020000  002604                          ^O020000,CALC
010007  106650  000064  002764                  CMP     U.CCYL(R5),CYL  , SEE IF LO CYL MATCHES
010012                                          BNE     7$              ; IF NOT, BRANCH
010012  050000  010245                          ^O050000,7$
010014  106650  000065  002765                  CMP     U.CCYL+1(R5),CYL+1      ; SEE IF HI CYL MATCHES
010017                                          BNE     7$              ; IF NOT, BRANCH
010017  050000  010245                          ^O050000,7$
010021  106650  000066  002766                  CMP     U.CGRP(R5),GROUP        ; SEE IF GROUP MATCHES
010024                                          BNE     7$              ; IF NOT, BRANCH
010024  050000  010245                          ^O050000,7$
```

```
                                        .SBTTL  LINK   - BUILD A LINK (NODE) IN THE READ/WRITE CHAIN
                                        ;LINK
                                        ;
                                        ;       LINK WILL CREATE THE LINKS FOR THE CHAIN
                                        ;
 6 010026  104653  000046                      MOV     U.PARM(R5),R3   ; R3 HAS UNIT PARAMETERS
 7 010030  104307  003015                      MOV     CHAINS,R0       ; R0 POINTS TO START OF CHAIN
 8 010032                                      BNE     22$             ; IF THE CHAIN IS ALLREADY STARTED, BRANCH
   010032  050000  010062                      ^050000,22$
 9 010034  102203  000100                      BIT     #REDWRT,R3      ; SEE IF READ IN PROGRESS
10 010036                                      BEQ     21$             ; IF SO, BRANCH
   010036  010000  010052                      ^010000,21$
11 010040  102203  000400                      BIT     #REVEC,R3       ; SEE IF TRYING TO FIND A REPLACEMENT
12 010042                                      BNE     21$             ; IF SO, YOU'RE TRYING TO READ, SO BRANCH
   010042  050000  010052                      ^050000,21$
13 010044  104207  000401                      MOV     #WBUFLN,R0      ; R0 HAS NUMBER OF WORDS IN WRITE BUFFER
14 010046                                      CALL    ALLOCM          ; ALLOCATE THE MEMORY
   010046  020000  010271                      ^020000,ALLOCM
15 010050  104070  010300                      MOV     R0,SECPTR       ; SECPTR POINTS TO WRITE BUFFER
16 010052  104207  000010           21$:       MOV     #LINKLN,R0      ; R0 HAS LENGTH OF CHAIN LINK
17 010054                                      CALL    ALLOCM          ; ALLOCATE THE MEMORY
   010054  020000  010271                      ^020000,ALLOCM
18 010056  104070  003015                      MOV     R0,CHAINS       ; CHAINS POINTS TO FIRST LINK
19 010060                                      BR      24$             ; BRANCH
   010060  000000  010105                      ^00,24$
20 010062  104171                   22$:       MOV     (R0),R1         ; R1 HAS NEXT LINK POINTER AND STATUS
29 010063                                      ASSUME  RW.CPT,0        ; ASSUME STATUS AND POINTER FIRST WORD
33 010063  102201  100000                      BIT     #EOC,R1         ; SEE IF THIS LINK IS THE LAST
34 010065                                      BNE     23$             ; IF SO, BRANCH
   010065  050000  010074                      ^050000,23$
35 010067  103201  140000                      BIC     #UNADDR,R1      ; CLEAR UNUSED ADDRESSING BITS
36 010071  104017                              MOV     R1,R0           ; R0 POINTS TO NEXT LINK
37 010072                                      BR      22$             ; LOOP
   010072  000000  010062                      ^00,22$
38 010074  104072                   23$:       MOV     R0,R2           ; SAVE R0
39 010075  104207  000010                      MOV     #LINKLN,R0      ; R0 HAS LENGTH OF LINK
40 010077                                      CALL    ALLOCM          ; ALLOCATE MEMORY FOR LINK
   010077  020000  010271                      ^020000,ALLOCM
41 010101  103201  100000                      BIC     #EOC,R1         ; CLEAR THE END-OF-CHAIN FLAG
42 010103  101071                              BIS     R0,R1           ; PUT IN POINTER TO NEXT LINK
43 010104  100121                              MOV     R1,(R2)         ; PUT POINTER BACK IN LAST LINK
52 010105                                      ASSUME  RW.CPT,0        ; ASSUME STATUS AND POINTER FIRST WORD
56 010105                           24$:
```

```
                                .SBTTL  FILLIN - FILL THE READ/WRITE CHAIN LINK (NODE) WITH REQUIRED INFORMATION
                                ;FILLIN
                                ;
                                ;       FILLIN BUILDS THE PARAMETERS THE UDA REQUIRES TO WRITE OR READ A BLOCK
                                ;
  010105  104302  002767        MOV     TRACK,R2        ; GET TRACK (HEAD) NUMBER
  010107  102203  000100        BIT     #REDWRT,R3      ; SEE IF READ IN PROGRESS
  010111                        BEQ     11$             ; IF SO, BRANCH
  010111  010000  010131        ^O10000,11$
  010113  102203  000400        BIT     #REVEC,R3       ; SEE IF TRYING TO FIND A REPLACEMENT
  010115                        BNE     11$             ; IF SO, YOU'RE TRYING TO READ, SO BRANCH
  010115  050000  010131        ^O50000,11$
  010117  101202  122400        BIS     #WREAL,R2       ; BUILD WRITE REAL TIME COMMAND
  010121  100672  000005        MOV     R2,RW.CMD(R0)   ; MOVE TO SDI REAL TIME COMMAND AREA
  010123  104202  140000        MOV     #WSTOP,R2       ; MOVE LAST BLOCK FLAG TO R2
  010125  104301  010300        MOV     SECPTR,R1       ; R1 POINTS TO WRITE BUFFER
  010127                        BR      12$             ; BRANCH
  010127  000000  010146        ^O00,12$
  010131  101202  013400  11$:  BIS     #RREAL,R2       ; BUILD READ REAL TIME COMMAND
  010133  100672  000005        MOV     R2,RW.CMD(R0)   ; MOVE TO SDI REAL TIME COMMAND AREA
  010135  104202  100000        MOV     #RSTOP,R2       ; MOVE LAST BLOCK FLAG TO R2
  010137                        PUSH    R0              ; SAVE R0
  010137  100467                                                          MOV R0,-(SP)
  010140  104207  000415        MOV     #RBUFLN,R0      ; SIZE OF READ BUFFER
  010142                        CALL    ALLOCM          ; ALLOCATE BUFFER
  010142  020000  010271        ^O20000,ALLOCM
  010144  104071                MOV     R0,R1           ; R1 POINTS TO BUFFER
  010145                        POP     R0              ; RESTORE R0
  010145  104267                                                          MOV (SP)+,R0
  010146  100172  12$:  MOV     R2,(R0)         ; MOVE LAST BLOCK FLAG TO NEXT BLOCK POINTER
  010147                        ASSUME  RW.CPT,0
  010147  100671  000002        MOV     R1,RW.BUF(R0)   ; MOVE POINTER TO SECTOR TO POINTER AREA
  010151  104202  002772        MOV     #DUMSDI,R2      ; R2 POINTS TO DUMMY SDI AREA
  010153  100672  000006        MOV     R2,RW.SDI(R0)   ; MOVE R2 TO POINTER TO DUMMY SDI AREA
  010155  104302  002770        MOV     SECTOR,R2       ; R2 CONTAINS SECTOR TO BE WRITTEN
  010157  100672  000007        MOV     R2,RW.ANG(R0)   ; MOVE TO ANGLE FROM INDEX
  010161  104302  002760        MOV     CURBN,R2        ; MOVE LOW ORDER BN TO R2
  010163  100672  000003        MOV     R2,RW.LOW(R0)   ; MOVE LOW ORDER BN TO LOW EXPECTED HEADER
  010165  104302  002761        MOV     CURBN+1,R2      ; MOVE HIGH ORDER BN TO R2
  010167  102203  000200        BIT     #RBNBN,R3       ; SEE IF BN IS AN RBN
  010171                        BNE     15$             ; IF SO, BRANCH
  010171  050000  010204        ^O50000,15$
  010173  104143                MOV     (R4),R3         ; GET SUBUNIT PARAMETERS
  010174                        ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
  010174  102203  020000        BIT     #DCYLS,R3       ; SEE IF USING DIAGNOSTIC CYLINDERS
  010176                        BEQ     16$             ; IF NOT, BRANCH
  010176  010000  010206        ^O10000,16$
  010200  101202  140000        BIS     #HD.DBN,R2      ; DIAGNOSTIC HEADER
  010202                        BR      16$             ; BRANCH
  010202  000000  010206        ^O00,16$
  010204  101202  060000  15$:  BIS     #HD.RBN,R2      ; REVECTORED HEADER
  010206  105302  002755  16$:  ADD     HIBN,R2         ; ADD HI BN BITS
  010210                        ASSUME  HD.LBN,0
  010210  100672  000004        MOV     R2,RW.HI(R0)    ; MOVE R2 TO HI WORD EXPECTED HEADER
  010212  104302  002771        MOV     INDEX,R2        ; GET DISTANCE FROM INDEX
  010214  115402                INC     R2              ; ADJUST INDEX
  010215  106302  003002        CMP     SCR1,R2         ; SEE IF FIRST SECTOR AFTER INDEX
```

```
 60 010217                                          BNE     13$             ; IF NOT, BRANCH
    010217  050000  010222                         ^050000,13$
 61 010221  114002                                  CLR     R2              ; ZERO THETA FROM INDEX
 62 010222  100672  000007          13$:    MOV     R2,RW.ANG(R0)   ; SAVE
 66 010224  114002                                  CLR     R2              ; ZERO R2
 67 010225  100672  000001                         MOV     R2,RW.STAT(R0)  ; ZERO STATUS
 71 010227  107200  000001  003020                 SUB     #1,MAXSEC       ; DECREMENT SECTOR COUNT
 72 010232                                          BEQ     7$              ; IF COUNT COMPLETE, BRANCH
    010232  010000  010245                         ^010000,7$
 73 010234  105200  000001  002760                 ADD     #1,CURBN        ; ADD ONE TO LO CURRENT SECTOR
 74 010237                                          BCC     5$              ; IF NO CARRY, BRAN~ 1
    010237  040000  010005                         ^040000,5$
 75 010241  115400  002761                         INC     CURBN+1         ; PROPOGATE CARRY
 76 010243                                          BR      5$              ; BRANCH
    010243  000000  010005                         ^00,5$
 77 010245  114001                  7$:     CLR     R1              ; IMMIDATE CALL
 78 010246  114002                                  CLR     R2              ; NO ERRORS
 79 010247  104653  000046                         MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
 80 010251  102203  000100                         BIT     #REDWRT,R3      ; SEE IF READ IS TO BE DONE
 81 010253                                          BEQ     2$              ; IF SO, BRANCH
    010253  010000  010265                         ^010000,2$
 82 010255  102203  000400                         BIT     #REVEC,R3       ; SEE IF READING RCT
 83 010257                                          BNE     2$              ; IF SO, BRANCH
    010257  050000  010265                         ^050000,2$
 84 010261  104207  010301                         MOV     #WRITE,R0       ; WRITE ROUTINE IS NEXT
 85 010263                                          BR      3$              ; BRANCH
    010263  000000  010267                         ^00,3$
 86 010265  104207  011615          2$:     MOV     #READ,R0        ; READ ROUTINE IS NEXT
 87 010267                          3$:     BR      JMPRET
    010267  000000  003746                         ^00,JMPRET
 88                                                 .DSABL  LSB
```

```
                                        .SBTTL  ALLOCM - ALLOCATE MEMORY FOR THE READ/WRITE BUFFERS AND CHAIN
010271                                  ALLOCM:
                                        ;
                                        ;       ALLOCATE MEMORY FOR READ/WRITE BUFFERS
                                        ;
010271  107070  010277                  SUB     R0,TMEMPL       ; SUBTRACT LENGTH FROM MEMORY POOL
010273  104307  010277                  MOV     TMEMPL,R0       ; R0 POINTS TO MEMORY ALLOCATED
010275                                  RETURN                  ; RETURN TO CALLING MODULE
010275  000000  000000                  ^O0,0

010277                                  TMEMPL: .BLKW   1               ; TEMPORARY MEMORY POOL FOR READ/WRITE
010300                                  SECPTR: .BLKW   1               ; SECTOR POINTER FOR WRITE
```

```
      1                                 .SBTTL  ********** OVERLAY MODULE WRITE
      9                                 ;*****************************************************************************
     10                                 ;*****************************************************************************
     11                                 ;*****************************************************************************
     12                                 ;*****************************************************************************
     13                                 ;
     14                                 ;
     15                                 ;
     24 010301                          WRITE:
     28                                 ;
     29                                 ;       WRITE GENERATES THE PATTERNS AND WRITES THEM TO THE DRIVE
     30                                 ;
     38                                         .ENABL  LSB
     39 010301                                  CALL    DSABLE          ; DISABLE ERROR RECOVERY
        010301  020000  005046                  ^O020000,DSABLE
     40 010303  104657  000012                  MOV     U.RWTO(R5),R0   ; MOVE READ/WRITE TIMEOUT VALUE TO R0
     41 010305  115407                          INC     R0              ; INCREMENT COUNT
     42 010306  100657  000012                  MOV     R0,U.RWTO(R5)   ; SAVE
     43 010310                                  BEQ     8$              ; IF ZERO, WE KNOW THAT THE TIMEOUT IS UNEX
        010310  010000  010407                  ^O010000,8$
     44 010312  106207  000002                  CMP     #2,R0           ; SEE IF TRIED MAXIMUM TIMES
     45 010314                                  BCS     11$             ; IF SO, BRANCH
        010314  040000  010320                  ^O040000,..+3
        010316  000000  010325                  ^O00,11$
     46 010320  104141                          MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
     47 010321                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
     48 010321  102201  001000                  BIT     #RTRIES,R1      ; SEE IF RETRIES ARE ENABLED
     49 010323                                  BNE     8$              ; IF SO, BRANCH
        010323  050000  010407                  ^O050000,8$
     50 010325                          11$:    HARDER  25,<S.LETR(R4),U.CBN(R5),U.CBN+1(R5)>
        010325  104200  002337  001601                                  MOV     #ER25,HRQ.04
        010330  104640  000005  001602                                  MOV     S.LETR(R4),HRQ.05
        010333  104650  000053  001603                                  MOV     U.CBN(R5),HRQ.06
        010336  104650  000054  001604                                  MOV     U.CBN+1(R5),HRQ.07
        010341  104202  107671                                          MOV     #25!ERHARD+4000.,R2
        010343  104020  001577                                          MOV     R2,HRQ.02
        010345  104200  010345  001576                                  MOV     #.,HRQ.01
        010350  104200  060014  001575                                  MOV     #ERRMC,HRQ.RQ
     51 010353                                  ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5),U.RBN+1(R5)>
        010353  104650  000046  001605                                  MOV     U.PARM(R5),HRQ.08
        010356  104200  005472  001606                                  MOV     #RBNTXT,HRQ.09
        010361  104650  000055  001607                                  MOV     U.RBN(R5),HRQ.10
        010364  104650  000056  001610                                  MOV     U.RBN+1(R5),HRQ.11
     52 010367                                  ENDERR  0
        010367  114000  003013                                          CLR     ERRPOS          ; CLEAR THE POSITION
     53 010371  104653  000046                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
     54 010373  103203  000200                  BIC     #RBNBN,R3       ; IF HANDLING AN RBN, CLEAR IT
     55 010375  100653  000046                  MOV     R3,U.PARM(R5)   ; SAVE
     56 010377  104201  000001                  MOV     #1,R1           ; ONLY 1 SECTOR HANDLED
     57 010401  100651  000021                  MOV     R1,U.NSEC(R5)   ; STORE
     58 010403  104207  005134                  MOV     #SETUP,R0       ; SETUP IS NEXT MODULE CALLED
     59 010405                                  BR      7$              ; BRANCH
        010405  000000  010546                  ^O00,7$
     60 010407                          8$:     CALL    BULDUM          ; BUILD DUMMY SDI CONTROL BLOCK
        010407  020000  002552                  ^O020000,BULDUM
     61
     62                                 .SBTTL  BULDSC - BUILD THE SECTOR TO WRITE (FILL WITH PATTERN AND CALC EDC)
```

```
                                ;BULDSC
                                ;
                                ;       BULDSC BUILDS THE INFORMATION AND DATA PATTERN TO BE WRITTEN ON THE
                                ;       SECTOR
                                ;
 010411  104651  000014         MOV     U.PAT(R5),R1    ; GET PATTERN NUMBER TO WRITE
 010413  104013                 MOV     R1,R3           ; COPY PATTERN NUMBER TO R3
 010414  110701                 SWAB    R1              ; SWAP THE BYTES
 010415  101013                 BIS     R1,R3           ; REPLICATE THE PATTERN NUMBER
 010416  104031                 MOV     R3,R1           ; COPY TO R1
 010417  110201                 ROL     R1              ; ROTATE TO NEXT POSITION
 010420  110201                 ROL     R1
 010421  110201                 ROL     R1
 010422  110201                 ROL     R1
 010423  103201  007417         BIC     #^CHBLONB!^CLBLONB,R1   ; CLEAR UNUSED BITS
 010425  101013                 BIS     R1,R3           ; REPLICATE THE PATTERN
 010426  104307  003015         MOV     CHAINS,R0       ; GET POINTER TO FIRST WRITE CHAIN NODE
 010430  104677  000002         MOV     RW.BUF(R0),R0   ; R0 POINTS TO BUFFER AREA
 010432  100273                 MOV     R3,(R0)+        ; MOVE PATTERN NUMBERS TO SECTOR
                                .SBTTL  COPPAT - COPY THE DATA PATTERN TO BUFFER
                                ;COPPAT
                                ;
                                ;       COPPAT COPIES THE PATTERN TO THE SECTOR TO BE WRITTEN
                                ;
 010433                         PUSH    <R0,R4,R5>      ; SAVE R0,R4,R5
 010433  100467                                                         MOV R0,-(SP)
 010434  100464                                                         MOV R4,-(SP)
 010435  100465                                                         MOV R5,-(SP)
 010436  104652  000014             MOV     U.PAT(R5),R2    ; R2 HAS PATTERN NUMBER
 010440  104622  003021             MOV     PATPTR(R2),R2   ; POINT TO PATTERN
 010442  115402                     INC     R2              ; SKIP EDC
 010443  104201  000377             MOV     #SCTWRD,R1      ; R1 HAS NUMBER OF WORDS TO FILL WITH PATTERN
 010445  104024             COPLP0: MOV     R2,R4           ; R4 POINTS TO LENGTH OF PATTERN
 010446  104243                     MOV     (R4)+,R3        ; R3 CONTAINS LENGTH OF PATTERN
 010447  106203  000001             CMP     #1,R3           ; SEE IF PATTERN IS 1 WORD LONG
 010451                           BEQ     ONEPAT          ; IF SO, BRANCH
 010451  010000  010465           ^O10000,ONEPAT
 010453  104245             COPLP1: MOV     (R4)+,R5        ; R5 GETS 1 WORD OF THE DATA PATTERN
 010454  100275                     MOV     R5,(R0)+        ; MOVE PATTERN WORD TO SECTOR AREA
 010455  117401                     DEC     R1              ; DECREMENT NUMBER OF WORDS TO WRITE IN SECTOR
 010456                           BEQ     COPFIN          ; IF ALL WORDS WRITTEN, BRANCH
 010456  010000  010472           ^O10000,COPFIN
 010460  117403                   DEC     R3              ; DECREMENT COUNT OF WORDS IN PATTERN
 010461                           BNE     COPLP1          ; IF DATA PATTERN UNFINISHED, BRANCH
 010461  050000  010453           ^O50000,COPLP1
 010463                           BR      COPLP0          ; BRANCH
 010463  000000  010445           ^O0,COPLP0
 010465  104145             ONEPAT: MOV     (R4),R5         ; GET 1 WORD OF DATA PATTERN
 010466  100275             COPLP2: MOV     R5,(R0)+        ; MOVE PATTERN TO SECTOR AREA
 010467  117401                     DEC     R1              ; DECREMENT NUMBER OF WORDS TO MOVE TO SECTOR
 010470                           BNE     COPLP2          ; IF INCOMPLETE, BRANCH
 010470  050000  010466           ^O50000,COPLP2
 010472  104425             COPFIN: MOV     -(R2),R5        ; GET EDC
 010473  100175                     MOV     R5,(R0)         ; MOVE TO BUFFER
 010474                           POP     <R5,R4,R0>      ; RESTORE R5,R4,R0
 010474  104265                                                         MOV (SP)+,R5
 010475  104264                                                         MOV (SP)+,R4
```

```
                                                                                    MOV (SP)+,R0
    010476  104267
110 010477  104651  000014                      MOV     U.PAT(R5),R1    ; GET PATTERN NUMBER
111 010501                                      BNE     21$             ; IF NOT PATTERN 16, BRANCH
    010501  050000  010510                      ^050000,21$
112 010503  117407                              DEC     R0              ; POINT TO START OF BUFFER
113 010504                                      CALL    CMPEDC          ; COMPUTE EDC
    010504  020000  001640                      ^020000,CMPEDC
114 010506  100672  000400                      MOV     R2,BF.EDC(R0)   ; SAVE EDC
115 010510                          21$:        CALL    WBLOCK          ; WRITE THE SECTOR
    010510  020000  010550                      ^020000,WBLOCK
116 010512  115002                              TST     R2              ; SEE IF ERROR OCCURRED
117 010513                                      BNE     7$              ; IF NOT, BRANCH
    010513  050000  010546                      ^050000,7$
118 010515  104643  000011                      MOV     S.MEGW(R4),R3   ; GET MEGABYTE COUNT
119 010517  106203  003642                      CMP     #1954.,R3       ; SEE IF ONE MEGABYTE TRANSFERED
120 010521                                      BPL     6$              ; BRANCH IF NOT
    010521  030000  010542                      ^030000,6$
121 010523  107203  003642                      SUB     #1954.,R3       ; CLEAR MEGABYTE COUNT
122 010525  100643  000011                      MOV     R3,S.MEGW(R4)   ; SAVE NEW COUNT
123 010527  114000  001577                      CLR     HRQ.02          ; NOT REPORTING READS
124 010531  104200  000001  001600              MOV     #1,HRQ.03       ; REPORT 1 MEGABIT WRITTEN
125 010534  104202  060011                      MOV     #T4MXFR,R2      ; SET UP FOR MEGABIT REPORT
126 010536  104020  001575                      MOV     R2,HRQ.RQ       ; FLAG AS NON-ERROR
127 010540                                      BR      1$              ; EXIT
    010540  000000  010543                      ^00,1$
128 010542  114002                      6$:     CLR     R2              ; NO ERRORS
129 010543  104207  010744              1$:     MOV     #AFTWRT,R0      ; AFTWRT IS NEXT MODULE
130 010545  114001                      2$:     CLR     R1              ; IMMIDATE CALL
131 010546                              7$:     BR      JMPRET          ; RETURN TO RDWRT
    010546  000000  003746                      ^00,JMPRET
132                                             .DSABL  LSB
```

```
                                .SBTTL  WBLOCK - WRITE THE SECTOR(S)
                                WBLOCK:
                                ;
                                ;       WBLCCK WRITES THE SECTOR(S) ONTO THE DRIVE
                                ;
 010550                         ;       PUSH    R4              ; SAVE R4
 010550  100464                                                                  MOV R4,-(SP)
 010551  104307  003015         ;       MOV     CHAINS,R0       ; POINT TO START OF CHAIN
 010553  104652  000025         ;       MOV     U.MASK(R5),R2   ; R2 HAS SDI INTERCONNECT
 010555  104644  000007         ;       MOV     S.SCHR(R4),R4   ; R4 POINTS TO SUBUNIT CHARACTERISTICS
 010557  104644  000005         ;       MOV     DATPRE(R4),R4   ; R4 CONTAINS DATA PREAMBLE LENGTH
 010561  103204  177400         ;       BIC     #HIBYTE,R4      ; STRIP OFF UNUSED BITS
 010563  060012                 ;       XFC     WAITSI          ; WAIT FOR SECTOR OR INDEX PULSE
 010564  115001                 ;       TST     R1              ; SEE IF ERROR OCCURRED
 010565                         ;       BEQ     3$
 010565  010000  010630         ;       ^010000,3$
 010567                         ;       DEVFTL  42,<U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
 010567  104200  003066  001601                                                  MOV     #FR42,HRQ.04
 010572  104650  000066  001602                                                  MOV     U.CGRP(R5),HRQ.05
 010575  104650  000064  001603                                                  MOV     U.CCYL(R5),HRQ.06
 010600  104650  000065  001604                                                  MOV     U.CCYL+1(R5),HRQ.07
 010603  104202  047712                                                          MOV     #42!FTLDEV+4000.,R2
 010605  104020  001577                                                          MOV     R2,HRQ.02
 010607  104200  010607  001576                                                  MOV     #.,HRQ.01
 010612  104200  060014  001575                                                  MOV     #ERRMC,HRQ.RQ
 010615                         ;       ENDERR
 010615  104200  000051  001605                                                  MOV     #SER22,HRQ.08
 010620  104200  000011  003013                                                  MOV     #8+1,ERRPOS     ; SET THE POSITION
 010623                         ;       CALL    GOSEEK
 010623  020000  005066         ;       ^020000,GOSEEK
 010625                         ;       POP     R4              ; RESTORE SUBUNIT POINTER
 010625  104264                                                                 MOV (SP)+,R4
 010626                         ;       BR      6$              ; EXIT
 010626  000000  010742         ;       ^00,6$
 010630                         3$:
 010630  104202  000400         ;       MOV     #SEC512,R2      ; WRITE A 512 BYTE SECTOR
 010632  060003                 ;       XFC     XWRITE          ; WRITE THE SECTOR(S)
 010633                         ;       POP     R4              ; RESTORE R4
 010633  104264                                                                 MOV (SP)+,R4
 010634  100651  000061         ;       MOV     R1,U.RWER(R5)   ; SAVE ERROR IF ANY
 010636                         ;       BEQ     5$              ; IF NO ERROR, BRANCH
 010636  010000  010712         ;       ^010000,5$
 010640  104651  000026         ;       MOV     U.WRIT(R5),R1   ; GET WRITE PROTECTION STATUS
                                ;       NOW I OR IN THE READ-ONLY WRITE PROTECT BITS.  THIS WILL CATCH
                                ;       IT IF SOMEHOW I GET INTO THIS MODULE ON A READ-ONLY DRIVE
                                ;       (I HOPE NOT)
 010642  101651  000045         ;       BIS     U.WPRT(R5),R1   ; SET READ-ONLY BITS
 010644  104652  000050         ;       MOV     U.SUBU(R5),R2   ; GET SUBUNIT IN USE
 010646  105202  000003         ;       ADD     #3,R2           ; ADD 3 TO SHIFT WRITE PROT STAT TO LO NIBBLE
 010650  110601                 10$:    ROR     R1              ; ROTATE PROTECTION STATUS (LO BIT TO CARRY)
 010651  117402                 ;       DEC     R2              ; DECREMENT SUBUNIT IN USE
 010652                         ;       BPL     10$             ; IF >= 0, BRANCH
 010652  030000  010650         ;       ^030000,10$
 010654  110601                 ;       ROR     R1              ; MOVE THE BIT INTO CARRY TO TEST
 010655                         ;       BCC     7$              ; IF CARRY CLEAR (NOT WRITE PROTECTED) BRANCH
 010655  040000  010741         ;       ^040000,7$
 010657                         ;       DEVFTL  18,U.RWER(R5)   ; REPORT WRITE ON WRITE PROTECTED DRIVE
```

```
        010657  104200  001702  001601                                                  MOV     #ER18,HRQ.04
        010662  104650  000061  001602                                                  MOV     U.RWER(R5),HRQ.05
        010665  104202  047662                                                          MOV     #18!FTLDEV+4000.,R2
        010667  104020  001577                                                          MOV     R2,HRQ.02
        010671  104200  010671  001576                                                  MOV     #.,HRQ.01
        010674  104200  060014  001575                                                  MOV     #ERRMC,HRQ.RQ
    45  010677                          ENDERR
        010677  104200  000051  001603                                                  MOV     #SER22,HRQ.06
        010702  104200  000007  003013                                                  MOV     #6+1,ERRPOS     ; SET THE POSITION
    46  010705                          CALL    GOSEEK
        010705  020000  005066          ^O20000,GOSEEK
    47  010707  104051                  MOV     R5,R1           ; DEFFERED CALL
    48  010710                          BR      6$              ; BRANCH
        010710  000000  010742          ^O0,6$
    49  010712  104657  000012  5$:     MOV     U.RWTO(R5),R0   ; GET TIMEOUTS
    50  010714                          BEQ     7$              ; IF NO RETRIES, BRANCH
        010714  010000  010741          ^O10000,7$
    51  010716                          REPSFT  SOFT            ; REPORT ONE SOFT ERROR
        010716  104200  000001  001577                                                  MOV     #1,HRQ.02
        010721  114000  001600                                                          CLR     HRQ.03
        010723  114000  001601                                                          CLR     HRQ.04
        010725  100467                                                                          MOV R0,-(SP)
        010726  104657  000063                                                          MOV     U.UNUM(R5),R0
        010730  105657  000050                                                          ADD     U.SUBU(R5),R0
        010732  104070  001576                                                          MOV     R0,HRQ.01
        010734  104207  060007                                                          MOV     #T4SOFT,R0
        010736  020000  001543          ^O20000,HOSTRQ
        010740  104267                                                                          MOV (SP)+,R0
    52  010741  114002          7$:     CLR     R2              ; NO ERRORS
    53  010742                  6$:     RETURN                  ; RETURN TO CALLING PROGRAM;
        010742  000000  000000          ^O0,0
```

```
                                .SBTTL  ********** OVERLAY MODULE AFTWRT
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                AFTWRT:
                                ;
                                ;       AFTWRT WILL HANDLE ANY ERRORS THAT OCCURRED DURING THE WRITE
                                ;       OPERATION
                                ;
                                        .ENABL  LSB
                                .SBTTL  FNDWER - IF ERROR DURING WRITE, FIND IT'S POSITION IN THE CHAIN
                                ;FNDWER
                                ;
                                ;       FIND THE FIRST BUFFER THAT IS NOT WRITTEN
                                ;
010744                                  PUSH    <R4,R5>         ; SAVE REGISTERS
010744  100464                                                                  MOV R4,-(SP)
010745  100465                                                                  MOV R5,-(SP)
010746  104654  000053                 MOV     U.CBN(R5),R4    ; GET LO ORDER BN
010750  104655  000054                 MOV     U.CBN+1(R5),R5  ; GET HI BN
010752  114002                          CLR     R2              ; CLEAR SECTOR COUNT
010753  104307  003015                 MOV     CHAINS,R0       ; R0 POINTS TO FIRST BUFFER
010755  100674  000003         31$:    MOV     R4,RW.LOW(R0)   ; MOVE TO CHAIN
010757  100675  000004                 MOV     R5,RW.HI(R0)    ; MOVE TO CHAIN
010761  104171                          MOV     (R0),R1         ; GET STATUS BITS
010762                                  ASSUME  RW.CPT,0
010762  102201  040000                 BIT     #BUFFLG,R1      ; SEE IF THIS BUFFER HAS BEEN WRITTEN
010764                                  BNE     32$             ; IF NOT, BRANCH
010764  050000  011004                 ^050000,32$
010766  115402                          INC     R2              ; INCREMENT SECTOR COUNT
010767  115001                          TST     R1              ; SEE IF END-OF-LIST
010770                                  BMI     32$             ; IF LAST BUFFER, EXIT
010770  070000  011004                 ^070000,32$
010772                                  ASSUME  EOC,100000      ; ASSUME WRITE STOP BIT IS SIGN BIT
010772  103201  140000                 BIC     #UNADDR,R1      ; CLEAR UNUSED ADDRESSING BITS
010774  104017                          MOV     R1,R0           ; MOVE TO R0
010775  105204  000001                 ADD     #1,R4           ; ADD ONE TO LOW ORDER BN
010777                                  BCC     31$             ; IF NO CARRY, BRANCH
010777  040000  010755                 ^040000,31$
011001  115405                          INC     R5              ; PRPOGATE CARRY
011002                                  BR      31$             ; LOOP
011002  000000  010755                 ^00,31$
011004                          32$:    POP     <R5,R4>         ; RESTORE REGISTERS
011004  104265                                                                  MOV (SP)+,R5
011005  104264                                                                  MOV (SP)+,R4
011006  100652  000021                 MOV     R2,U.NSEC(R5)   ; SAVE AS NUMBER OF SECTORS HANDLED
011010  105642  000011                 ADD     S.MEGW(R4),R2   ; ADD TO MEGABITS WRITTEN
011012  100642  000011                 MOV     R2,S.MEGW(R4)   ; SAVE
011014  104652  000061                 MOV     U.RWER(R5),R2   ; SEE IF ANY ERROR OCCURRED
011016                                  BNE     3$              ; IF SO, BRANCH
011016  050000  011033                 ^050000,3$
011020  104657  000046                 MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
011022  103207  000200                 BIC     #RBNBN,R0       ; CLEAR RBN BIT
```

```
     85 011024  100657  000046                    MOV     R0,U.PARM(R5)   ; SAVE
     86 011026  104207  005134                    MOV     #SETUP,R0       ; SETUP NEXT MODULE
     87 011030  104051                            MOV     R5,R1           ; DEFERRED CALL
     88 011031                                    BR      17$             ; EXIT
        011031  000000  011612                    ^O0,17$
     89 011033                          3$:       CALL    BLKCHK          ; SEE IF THIS IS A KNOWN BAD BLOCK
        011033  020000  002465                    ^O20000,BLKCHK
     90 011035                                    BCS     15$             ; IF NOT, BRANCH
        011035  040000  011041                    ^O40000,.+3
        011037  000000  011053                    ^O0,15$
     91 011041  104657  000021                    MOV     U.NSEC(R5),R0   ; GET NUMBER OF SECTORS WRITTEN
     92 011043  115407                            INC     R0              ; SKIP THIS SECTOR
     93 011044  100657  000021                    MOV     R0,U.NSEC(R5)   ; SAVE
     94 011046  104207  005134                    MOV     #SETUP,R0       ; SETUP NEXT MODULE
     95 011050  104051                            MOV     R5,R1           ; DEFERRED CALL
     96 011051                                    BR      17$             ; EXIT
        011051  000000  011612                    ^O0,17$
     97 011053  104651  000021          15$:      MOV     U.NSEC(R5),R1   ; GET SECTORS WRITTEN
     98 011055  105651  000024                    ADD     U.CSEC(R5),R1   ; ADD TO TOTAL SECTORS WRITTEN
     99 011057  100651  000024                    MOV     R1,U.CSEC(R5)   ; SAVE
    100 011061  104651  000021                    MOV     U.NSEC(R5),R1   ; GET SECTORS WRITTEN
    101 011063  105651  000053                    ADD     U.CBN(R5),R1    ; ADD TO LO ORDER CURRENT BN
    102 011065  100651  000053                    MOV     R1,U.CBN(R5)    ; SAVE
    103 011067                                    BCC     2$              ; IF NO CARRY, BRANCH
        011067  040000  011076                    ^O40000,2$
    104 011071  104651  000054                    MOV     U.CBN+1(R5),R1  ; GET HI ORDER CURRENT BN
    105 011073  115401                            INC     R1              ; PROPOGATE CARRY
    106 011074  100651  000054                    MOV     R1,U.CBN+1(R5)  ; SAVE
    107 011076  114001                  2$:       CLR     R1              ; SET UP FOR NO SECTORS WTRITTEN
    108 011077  100651  000021                    MOV     R1,U.NSEC(R5)   ; SAVE
    109 011101  104651  000061                    MOV     U.RWER(R5),R1   ; GET WRITE ERROR
    110 011103  106201  000003                    CMP     #3,R1           ; SEE IF REVECTOR
    111 011105                                    BEQ     1$              ; IF SO, BRANCH
        011105  010000  011153                    ^O10000,1$
    112 011107  106201  000002                    CMP     #2,R1           ; SEE IF FAILURE IN DRIVE
    113 011111                                    BNE     6$              ; IF NOT, BRANCH
        011111  050000  011134                    ^O50000,6$
    114 011113                                    SOFTER  6               ; REPORT
        011113  104200  000562  001601                                            MOV     #ER6,HRQ.04
        011116  104202  147646                                                    MOV     #6!ERSOFT+4000.,R2
        011120  104020  001577                                                    MOV     R2,HRQ.02
        011122  104200  011122  001576                                            MOV     #.,HRQ.01
        011125  104200  060013  001575                                            MOV     #ERRMES,HRQ.RQ
    115 011130                                    CALL    GOINIT
        011130  020000  005106                    ^O20000,GOINIT
    116 011132                                    BR      12$             ; BRANCH
        011132  000000  011511                    ^O0,12$
    117 011134  106201  000004          6$:       CMP     #4,R1           ; SEE IF HEADER COMPARE FAILURE
    118 011136                                    BNE     8$              ; IF NOT, BRANCH
        011136  050000  011267                    ^O50000,8$
    119 011140  104651  000046                    MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
    120 011142  104143                            MOV     (R4),R3         ; GET SUBUNIT PARAMETERS
    121 011143                                    ASSUME  S.PARM,0
    122 011143  102203  020000                    BIT     #DCYLS,R3       ; SEE IF USING DIAGNOSTIC CYLINDERS
    123 011145                                    BNE     7$              ; IF SO, BRANCH
        011145  050000  011160                    ^O50000,7$
    124 011147  102201  000200                    BIT     #RBNBN,R1       ; SEE IF WRITING A RBN
```

```
125 011151                                      BNE     7$              ; IF SO, BRANCH
    011151  050000  011160                     ^O050000,7$
126 011153  104207  015007          1$:         MOV     #REVCT,R0       ; REVECTOR NEXT MODULE
127 011155  114001                              CLR     R1              ; IMMIDATE CALL
128 011156                                      BR      17$             ; EXIT
    011156  000000  011612                      ^O00,17$
129 011160                          7$:         HARDER  5               ; REPORT HEADER COMPARE FAILURE
    011160  104200  000365  001601                                              MOV     #ER5,HRQ.04
    011163  104202  107645                                                      MOV     #5!ERHARD+4000.,R2
    011165  104020  001577                                                      MOV     R2,HRQ.02
    011167  104200  011167  001576                                              MOV     #.,HRQ.01
    011172  104200  060014  001575                                              MOV     #ERRMC,HRQ.RQ
130 011175                                      ERRORC  <S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
    011175  104640  000005  001602                                              MOV     S.LETR(R4),HRQ.05
    011200  104670  000003  001603                                              MOV     RW.LOW(R0),HRQ.06
    011203  104670  000004  001604                                              MOV     RW.HI(R0),HRQ.07
131 011206                                      ERRORC  <R1,#RBNTXT,U.RBN(R5)>
    011206  104010  001605                                                      MOV     R1,HRQ.08
    011210  104200  005472  001606                                              MOV     #RBNTXT,HRQ.09
    011213  104650  000055  001607                                              MOV     U.RBN(R5),HRQ.10
132 011216                                      ERRORC  <U.RBN+1(R5),RW.ANG(R0)>
    011216  104650  000056  001610                                              MOV     U.RBN+1(R5),HRQ.11
    011221  104670  000007  001611                                              MOV     RW.ANG(R0),HRQ.12
133 011224                                      ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
    011224  104670  000005  001612                                              MOV     RW.CMD(R0),HRQ.13
    011227  104650  000066  001613                                              MOV     U.CGRP(R5),HRQ.14
    011232  104650  000064  001614                                              MOV     U.CCYL(R5),HRQ.15
    011235  104650  000065  001615                                              MOV     U.CCYL+1(R5),HRQ.16
134 011240                                      ERRORC  <U.LGRP(R5),U.LCYL(R5),U.LCYL+1(R5)>
    011240  104650  000071  001616                                              MOV     U.LGRP(R5),HRQ.17
    011243  104650  000067  001617                                              MOV     U.LCYL(R5),HRQ.18
    011246  104650  000070  001620                                              MOV     U.LCYL+1(R5),HRQ.19
135 011251                                      ENDERR  0
    011251  114000  003013                                                      CLR     ERRPOS          ; CLEAR THE POSITION
136 011253  103201  000200                      BIC     #RBNBN,R1       ; NO LONGER HANDLING AN RBN
137 011255  100651  000046                      MOV     R1,U.PARM(R5)   ; SAVE
138 011257  104201  000001                      MOV     #1,R1           ; ONLY 1 SECTOR WRITTEN
139 011261  100651  000021                      MOV     R1,U.NSEC(R5)   ; SAVE
140 011263  104207  005134                      MOV     #SETUP,R0       ; SETUP NEXT MODULE
141 011265                                      BR      13$             ; EXIT AND REPORT
    011265  000000  011613                      ^O00,13$
142 011267  106201  000153          8$:         CMP     #153,R1         ; SEE IF POSITIONER ERROR
143 011271                                      BNE     22$             ; IF NOT, BRANCH
    011271  050000  011337                      ^O050000,22$
144 011273                                      REPSFT  ,,SEEK          ; REPORT SEEK ERROR
    011273  114000  001577                                                      CLR     HRQ.02
    011275  114000  001600                                                      CLR     HRQ.03
    011277  104200  000001  001601                                              MOV     #1,HRQ.04
    011302  100467                                                                      MOV R0,-(SP)
    011303  104657  000063                                                      MOV     U.UNUM(R5),R0
    011305  105657  000050                                                      ADD     U.SUBU(R5),R0
    011307  104070  001576                                                      MOV     R0,HRQ.01
    011311  104207  060007                                                      MOV     #T4SOFT,R0
    011313  020000  001543                      ^O020000,HOSTRQ
    011315  104267                                                                      MOV (SP)+,R0
145 011316                                      SOFTER  44              ; POSITIONER ERROR
    011316  104200  003127  001601                                              MOV     #ER44,HRQ.04
```

```
        011321  104202  147714                                          MOV     #44!ERSOFT+4000.,R2
        011323  104020  001577                                          MOV     R2,HRQ.02
        011325  104200  011325  001576                                  MOV     #.,HRQ.01
        011330  104200  060013  001575                                  MOV     #ERRMES,HRQ.RQ
    146 011333                          CALL    GORCLB          ; RECALIBRATE NEEDED
        011333  020000  005076          ^O20000,GORCLB
    147 011335                          BR      12$             ; BRANCH
        011335  000000  011511          ^O0,12$
    148 011337  106201  000213    22$:  CMP     #213,R1         ; SEE IF READ/WRITE FAILURE
    149 011341                          BNE     23$             ; IF NOT, BRANCH
        011341  050000  011362          ^O50000,23$
    150 011343                          SOFTER  48              ; READ/WRITE READY FAILURE
        011343  104200  003243  001601                                  MOV     #ER48,HRQ.04
        011346  104202  147720                                          MOV     #48!ERSOFT+4000.,R2
        011350  104020  001577                                          MOV     R2,HRQ.02
        011352  104200  011352  001576                                  MOV     #.,HRQ.01
        011355  104200  060013  001575                                  MOV     #ERRMES,HRQ.RQ
    151 011360                          BR      12$             ; ERROR EXIT
        011360  000000  011511          ^O0,12$
    152 011362  106201  000253    23$:  CMP     #253,R1         ; SEE IF DRIVE DATA OR CLOCK TIMEOUT
    153 011364                          BNE     21$             ; IF NOT, BRANCH
        011364  050000  011407          ^O50000,21$
    154 011366                          SOFTER  47              ; DRIVE CLOCK TIMEOUT
        011366  104200  003214  001601                                  MOV     #ER47,HRQ.04
        011371  104202  147717                                          MOV     #47!ERSOFT+4000.,R2
        011373  104020  001577                                          MOV     R2,HRQ.02
        011375  104200  011375  001576                                  MOV     #.,HRQ.01
        011400  104200  060013  001575                                  MOV     #ERRMES,HRQ.RQ
    155 011403                          CALL    GOINIT          ; INIT THE DRIVE TO RECOVER
        011403  020000  005106          ^O20000,GOINIT
    156 011405                          BR      12$             ; ERROR EXIT
        011405  000000  011511          ^O0,12$
    157 011407  106201  000313    21$:  CMP     #313,R1         ; SEE IF RECIFVER READY TIMEOUT
    158 011411                          BNE     24$             ; IF NOT, BRANCH
        011411  050000  011434          ^O50000,24$
    159 011413                          SOFTER  49              ; RECEIVER READY FAILURE
        011413  104200  003270  001601                                  MOV     #ER49,HRQ.04
        011416  104202  147721                                          MOV     #49!ERSOFT+4000.,R2
        011420  104020  001577                                          MOV     R2,HRQ.02
        011422  104200  011422  001576                                  MOV     #.,HRQ.01
        011425  104200  060013  001575                                  MOV     #ERRMES,HRQ.RQ
    160 011430                          CALL    GOINIT          ; INIT THE DRIVE TO RECOVER
        011430  020000  005106          ^O20000,GOINIT
    161 011432                          BR      12$
        011432  000000  011511          ^O0,12$
    162 011434  106201  000413    24$:  CMP     #413,R1         ; SEE IF RTDS ERROR DURING WRITE
    163 011436                          BNE     25$             ; IF NOT, BRANCH
        011436  050000  011457          ^O50000,25$
    164 011440                          SOFTER  63              ; REPORT
        011440  104200  004115  001601                                  MOV     #ER63,HRQ.04
        011443  104202  147737                                          MOV     #63!ERSOFT+4000.,R2
        011445  104020  001577                                          MOV     R2,HRQ.02
        011447  104200  011447  001576                                  MOV     #.,HRQ.01
        011452  104200  060013  001575                                  MOV     #ERRMES,HRQ.RQ
    165 011455                          BR      12$             ; BRANCH
        011455  000000  011511          ^O0,12$
    166 011457                    25$:  HARDER  68,<#SER36,R1>          ; UNKNOWN ERROR CODE
```

```
                011457  104200  004175  001601                                          MOV     #ER68,HRQ.04
                011462  104200  004242  001602                                          MOV     #SER36,HRQ.05
                011465  104010  001603                                                  MOV     R1,HRQ.06
                011467  104202  107744                                                  MOV     #68!ERHARD+4000.,R2
                011471  104020  001577                                                  MOV     R2,HRQ.02
                011473  104200  011473  001576                                          MOV     #.,HRQ.01
                011476  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
    167 011501                          ENDERR
                011501  104200  000051  001604                                          MOV     #SER22,HRQ.07
                011504  104200  000010  003013                                          MOV     #7+1,ERRPOS     ; SET THE POSITION
    168 011507                          BR      14$
                011507  000000  011602          ^O0,14$
    169 011511                  12$:    CERROR  5,#SER19
                011511  104200  000470  001602                                          MOV     #SER19,HRQ.05
    170 011514                          ERRORC  <U.RWTO(R5),S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
                011514  104650  000012  001603                                          MOV     U.RWTO(R5),HRQ.06
                011517  104640  000005  001604                                          MOV     S.LETR(R4),HRQ.07
                011522  104670  000003  001605                                          MOV     RW.LOW(R0),HRQ.08
                011525  104670  000004  001606                                          MOV     RW.HI(R0),HRQ.09
    171 011530                          ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5)>
                011530  104650  000046  001607                                          MOV     U.PARM(R5),HRQ.10
                011533  104200  005472  001610                                          MOV     #RBNTXT,HRQ.11
                011536  104650  000055  001611                                          MOV     U.RBN(R5),HRQ.12
    172 011541                          ERRORC  <U.RBN+1(R5),RW.ANG(R0)>
                011541  104650  000056  001612                                          MOV     U.RBN+1(R5),HRQ.13
                011544  104670  000007  001613                                          MOV     RW.ANG(R0),HRQ.14
    173 011547                          ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
                011547  104670  000005  001614                                          MOV     RW.CMD(R0),HRQ.15
                011552  104650  000066  001615                                          MOV     U.CGRP(R5),HRQ.16
                011555  104650  000064  001616                                          MOV     U.CCYL(R5),HRQ.17
                011560  104650  000065  001617                                          MOV     U.CCYL+1(R5),HRQ.18
    174 011563                          ERRORC  <U.LGRP(R5),U.LCYL(R5),U.LCYL+1(R5)>
                011563  104650  000071  001620                                          MOV     U.LGRP(R5),HRQ.19
                011566  104650  000067  001621                                          MOV     U.LCYL(R5),HRQ.20
                011571  104650  000070  001622                                          MOV     U.LCYL+1(R5),HRQ.21
    175 011574                          ENDERR
                011574  104200  000051  001623                                          MOV     #SER22,HRQ.22
                011577  104200  000027  003013                                          MOV     #22+1,ERRPOS    ; SET THE POSITION
    176 011602                  14$:    CALL    GOSEEK          ; SEEK REQUIRED FOR ERROR RECOVERY
                011602  020000  005066          ^O020000,GOSEEK
    177 011604  104201  000001          MOV     #1,R1           ; DEFERRED CALL
    178 011606  100651  000022          MOV     R1,U.MSEC(R5)   ; ONLY TRY TO WRITE ONE SECTOR
    179 011610                          BR      13$             ; BRANCH
                011610  000000  011613          ^O0,13$
    180 011612  114002                  17$:    CLR     R2              ; NO ERRORS
    181 011613                  13$:    BR      JMPRET          ; RETURN TO CALLING PROGRAM
                011613  000000  003746          ^O0,JMPRET
    182                                 .DSABL  LSB
```

```
                                .SBTTL  ********** OVERLAY MODULE READ
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
 011615                         READ:
                                ;
                                ;       READ READS A BLOCK FROM THE DEVICE
                                ;
                                        .ENABL  LSB
 011615                                 CALL    DSABLE          ; DISABLE ERROR RECOVERY
 011615  020000  005046                 ^O020000,DSABLE
 011617  104657  000012                 MOV     U.RWTO(R5),R0   ; MOVE READ/WRITE TIMEOUT VALUE TO R0
 011621  115407                         INC     R0              ; INCREMENT COUNT
 011622  100657  000012                 MOV     R0,U.RWTO(R5)   ; SAVE READ/WRITE TIMEOUT
 011624                                 BEQ     5$              ; IF ZERO, FIRST TIME -- BRANCH
 011624  010000  011743                 ^O010000,5$
 011626  106207  000002                 CMP     #2,R0           ; SEE IF ATTEMPTED MAX TIMES
 011630                                 BCS     11$             ; IF SO, BRANCH
 011630  040000  011634                 ^O040000,..+3
 011632  000000  011647                 ^O0,11$
 011634  104141                         MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
 011635                                 ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
 011635  102201  001000                 BIT     #RTRIES,R1      ; SEE IF RETRIES ENABLED
 011637                                 BNE     5$              ; IF SO, BRANCH
 011637  050000  011743                 ^O050000,5$
 011641  104651  000046                 MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
 011643  102201  000400                 BIT     #REVEC,R1       ; SEE IF READING RCT
 011645                                 BNE     5$              ; IF SO, GO FOR RETRIES
 011645  050000  011743                 ^O050000,5$
 011647                         11$:    HARDER  26,<S.LETR(R4),U.CBN(R5),U.CBN+1(R5)>
 011647  104200  002366  001601                                         MOV     #ER26,HRQ.04
 011652  104640  000005  001602                                         MOV     S.LETR(R4),HRQ.05
 011655  104650  000053  001603                                         MOV     U.CBN(R5),HRQ.06
 011660  104650  000054  001604                                         MOV     U.CBN+1(R5),HRQ.07
 011663  104202  107672                                                 MOV     #26!ERHARD+4000.,R2
 011665  104020  001577                                                 MOV     R2,HRQ.02
 011667  104200  011667  001576                                         MOV     #.,HRQ.01
 011672  104200  060014  001575                                         MOV     #ERRMC,HRQ.RQ
 011675                                 ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5),U.RBN+1(R5)>
 011675  104650  000046  001605                                         MOV     U.PARM(R5),HRQ.08
 011700  104200  005472  001606                                         MOV     #RBNTXT,HRQ.09
 011703  104650  000055  001607                                         MOV     U.RBN(R5),HRQ.10
 011706  104650  000056  001610                                         MOV     U.RBN+1(R5),HRQ.11
 011711                                 ENDERR  0
 011711  114000  003013                                                 CLR     ERRPOS          ; CLEAR THE POSITION
 011713  104653  000046                 MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
 011715  103203  000200                 BIC     #RBNBN,R3       ; IF HANDLING AN RBN, CLEAR IT
 011717  100653  000046                 MOV     R3,U.PARM(R5)   ; SAVE
 011721  104207  000001                 MOV     #1,R0           ; ONE SECTOR HANDLED
 011723  100657  000021                 MOV     R0,U.NSEC(R5)   ; SAVE
 011725  104207  005134                 MOV     #SETUP,R0       ; SETUP IS NEXT MODULE CALLED
 011727  104051                         MOV     R5,R1           ; DELAYED CALL TO SETUP
 011730  102203  000400                 BIT     #REVEC,R3       ; SEE IF READING RCT
```

********** OVERLAY MODULE READ

```
     64 011732                                          BEQ     8$              ; IF NOT, BRANCH
        011732  010000  011752                         ^O10000,8$
     65 011734  104207  015007                         MOV     #REVCT,R0       ; REVECTOR NEXT MODULE
     66 011736  104200  177777  003003                 MOV     #-1,SCR2        ; MARK SECTOR AS BAD
     67 011741                                          BR      4$              ; EXIT
        011741  000000  011751                         ^O0,4$
     68 011743                                  5$:     CALL    BULDUM          ; BUILD THE DUMMY SDI CONTROL BLOCK
        011743  020000  002552                         ^O20000,BULDUM
     69 011745                                          CALL    RBLOCK          ; READ THE SECTOR
        011745  020000  011754                         ^O20000,RBLOCK
     70 011747  104207  012134                         MOV     #SECCHK,R0      ; SECCHK NEXT MODULE CALLED
     71 011751  114001                          4$:     CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
     72 011752                                  8$:     BR      JMPRET          ; RETURN TO RDWRT MODULE
        011752  000000  003746                         ^O0,JMPRET
     73                                                 .DSABL  LSB
```

```
                                        .SBTTL  RBLOCK - READ THE SECTORS
 2 011754                               RBLOCK:
                                        ;
                                        ;       READ A SECTOR FROM THE DEVICE
                                        ;
 6 011754  104307  003015               MOV     CHAINS,R0       ; POINT TO START OF CHAIN
 7 011756  104652  000025               MOV     U.MASK(R5),R2   ; R2 HAS SDI INTERCONNECT
 8 011760  060012                       XFC     WAITSI          ; WAIT FOR SECTOR OR INDEX PULSE
 9 011761  115001                       TST     R1              ; SEE IF ERROR OCCURRED
10 011762                               BEQ     6$
   011762  010000  012024               ^010000,6$
11 011764                               DEVFTL  42,<U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
   011764  104200  003066  001601                                        MOV     #ER42,HRQ.04
   011767  104650  000066  001602                                        MOV     U.CGRP(R5),HRQ.05
   011772  104650  000064  001603                                        MOV     U.CCYL(R5),HRQ.06
   011775  104650  000065  001604                                        MOV     U.CCYL+1(R5),HRQ.07
   012000  104202  047712                                               MOV     #42!FTLDEV+4000.,R2
   012002  104020  001577                                               MOV     R2,HRQ.02
   012004  104200  012004  001576                                        MOV     #.,HRQ.01
   012007  104200  060014  001575                                        MOV     #ERRMC,HRQ.RQ
12 012012                               ENDERR
   012012  104200  000051  001605                                        MOV     #SER22,HRQ.08
   012015  104200  000011  003013                                        MOV     #8+1,ERRPOS     ; SET THE POSITION
13 012020                               CALL    GOSEEK
   012020  020000  005066               ^020000,GOSEEK
14 012022                               BR      5$              ; EXIT
   012022  000000  012066               ^00,5$
15 012024                       6$:
19 012024  104202  000400               MOV     #SEC512,R2      ; READ A 512 BYTE SECTOR
23 012026  060002                       XFC     XREAD           ; WRITE THE SECTOR(S)
24 012027  100651  000061               MOV     R1,U.RWER(R5)   ; SAVE ERROR TYPE
25 012031                               CALL    FNDRER          ; FIND READ ERROR (FILL IN CORRECT BN NUMBERS)
   012031  020000  012070               ^020000,FNDRER
26 012033  115001                       TST     R1              ; SEE IF ANY ERRORS OCCURRED
27 012034                               BNE     4$              ; IF SO, BRANCH
   012034  050000  012065               ^050000,4$
28 012036  104657  000012               MOV     U.RWTO(R5),R0   ; SEE IF ANY RETRIES
29 012040                               BEQ     4$              ; IF SO, BRANCH
   012040  010000  012065               ^010000,4$
30 012042                               REPSFT  SOFT            ; REPORT SOFT ERROR
   012042  104200  000001  001577                                        MOV     #1,HRQ.02
   012045  114000  001600                                               CLR     HRQ.03
   012047  114000  001601                                               CLR     HRQ.04
   012051  100467                                                               MOV R0,-(SP)
   012052  104657  000063                                               MOV     U.UNUM(R5),R0
   012054  105657  000050                                               ADD     U.SUBU(R5),R0
   012056  104070  001576                                               MOV     R0,HRQ.01
   012060  104207  060007                                               MOV     #T4SOFT,R0
   012062  020000  001543               ^020000,HOSTRQ
   012064  104267                                                               MOV (SP)+,R0
31 012065  114002               4$:     CLR     R2              ; NO ERRORS
32 012066                       5$:     RETURN                  ; RETURN TO CALLING PROGRAM;
   012066  000000  000000               ^00,0
```

```
                                .SBTTL  FNDRER - IF ERROR DURING READ, FIND IT'S POSITION IN THE CHAIN
012070                          FNDRER:
                                ;
                                ;       FIND THE FIRST BUFFER THAT IS NOT READ
                                ;
012070                          PUSH    <R1,R4,R5>      ; SAVE REGISTERS
012070  100461                                                          MOV R1,-(SP)
012071  100464                                                          MOV R4,-(SP)
012072  100465                                                          MOV R5,-(SP)
012073  104654  000053          MOV     U.CBN(R5),R4    ; GET LO ORDER BN
012075  104655  000054          MOV     U.CBN+1(R5),R5  ; GET HI BN
012077  104307  003015          MOV     CHAINS,R0       ; R0 POINTS TO FIRST BUFFER
012101  100674  000003  7$:     MOV     R4,RW.LOW(R0)   ; MOVE TO CHAIN
012103  100675  000004          MOV     R5,RW.HI(R0)    ; MOVE TO CHAIN
012105  104171                  MOV     (R0),R1         ; GET STATUS BITS
012106  102201  040000          BIT     #BUFFLG,R1      ; SEE IF THIS BUFFER HAS BEEN READ
012110                          BEQ     8$              ; IF NOT, BRANCH
012110  010000  012127          ^O10000,8$
012112  115001                  TST     R1              ; SEE IF END-OF-LIST
012113                          BMI     8$              ; IF LAST BUFFER, EXIT
012113  070000  012127          ^O70000,8$
012115                          ASSUME  EOC,100000      ; ASSUME READ STOP IS SIGN
012115  103201  140000          BIC     #UNADDR,R1      ; CLEAR UNUSED BITS
012117  104017                  MOV     R1,R0           ; MOVE TO R0
012120  105204  000001          ADD     #1,R4           ; ADD ONE TO LOW ORDER BN
012122                          BCC     7$              ; IF NO CARRY, BRANCH
012122  040000  012101          ^O40000,7$
012124  115405                  INC     R5              ; PRPOGATE CARRY
012125                          BR      7$              ; LOOP
012125  000000  012101          ^O0,7$
012127                  8$:     POP     <R5,R4,R1>      ; RESTORE REGISTERS
012127  104265                                                          MOV (SP)+,R5
012130  104264                                                          MOV (SP)+,R4
012131  104261                                                          MOV (SP)+,R1
012132                          RETURN                  ; RETURN TO CALLING PROGRAM
012132  000000  000000          ^O0,0
```

```
                                .SBTTL  ********** OVERLAY MODULE SECCHK - CHECK THAT BUFFER FULL AND ECC
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                ;
012134                          SECCHK:
                                ;
                                ;       CHECK TO SEE IF THE SECTOR'S FULL, IF NOT, REPORT WHY (UNLESS
                                ;       REVECTOR, WHERE I'LL REVECTOR THE BLOCK)
                                ;       IF THE SECTOR'S FULL, CHECK ECC AND EDC, THEN CALL THE NEXT ROUTINE
                                ;       DEPENDING ON THE OPTIONS IN U.PARM (SELECTED BY USER)
                                ;
                                        .ENABL  LSB
012134  104307  003015                MOV     CHAINS,R0       ; GET POINTER TO READ CHAIN
012136  104171                        MOV     (R0),R1         ; GET STATUS OF BUFFER
012137                                 ASSUME  RW.CPT,0
012137  102201  040000                BIT     #BUFFLG,R1      ; SEE IF BUFFER FULL
012141                                 BNE     4$              ; IF SO, BRANCH
012141  050000  013010                ^O50000,4$
012143                                 CALL    BLKCHK          ; SEE IF THIS IS A KNOWN BAD BLOCK
012143  020000  002465                ^O20000,BLKCHK
012145                                 BCC     3$              ; IF SO, BRANCH
012145  040000  012774                ^O40000,3$
012147  104651  000021                MOV     U.NSEC(R5),R1   ; GET NUMBER OF SECTORS READ
012151  105651  000024                ADD     U.CSEC(R5),R1   ; ADD NUMBER OF SECTORS PREVIOUSLY HANDLED
012153  100651  000024                MOV     R1,U.CSEC(R5)   ; SAVE
012155  104651  000021                MOV     U.NSEC(R5),R1   ; GET NUMBER OF SECTORS READ
012157  105641  000010                ADD     S.MEGR(R4),R1   ; ADD TO MEGABITS READ
012161  100641  000010                MOV     R1,S.MEGR(R4)   ; SAVE
012163  104651  000021                MOV     U.NSEC(R5),R1   ; GET NUMBER OF SECTORS READ
012165  105651  000053                ADD     U.CBN(R5),R1    ; ADJUST U.CBN TO SECTOR WITH ERROR
012167  100651  000053                MOV     R1,U.CBN(R5)    ; SAVE
012171                                 BCC     1$              ; IF NO CARRY, BRANCH
012171  040000  012200                ^O40000,1$
012173  104651  000054                MOV     U.CBN+1(R5),R1  ; GET CURRENT BN
012175  115401                        INC     R1              ; PROPOGATE CARRY
012176  100651  000054                MOV     R1,U.CBN+1(R5)  ; SAVE
012200  114001                1$:     CLR     R1              ; TO CLEAR NUMBER OF SECTORS HANDLED
012201  100651  000021                MOV     R1,U.NSEC(R5)   ; ZERO SECTORS HANDLED
012203  104651  000061                MOV     U.RWER(R5),R1   ; GET READ/WRITE ERROR
012205  106201  000003                CMP     #3,R1           ; SEE IF REVECTORED BLOCK
012207                                 BEQ     2$              ; IF SO, BRANCH
012207  010000  012261                ^O10000,2$
012211  106201  000002                CMP     #2,R1           ; SEE IF FAILURE IN DRIVE
012213                                 BNE     8$              ; IF NOT, BRANCH
012213  050000  012236                ^O50000,8$
012215                                 SOFTER  20              ; REPORT
012215  104200  002056  001601                                        MOV     #ER20,HRQ.04
012220  104202  147664                                                MOV     #20!ERSOFT+4000.,R2
012222  104020  001577                                                MOV     R2,HRQ.02
012224  104200  012224  001576                                        MOV     #.,HRQ.01
012227  104200  060013  001575                                        MOV     #ERRMES,HRQ.RQ
012232                                 CALL    GOINIT
012232  020000  005106                ^O20000,GOINIT
```

********** OVERLAY MODULE SECCHK - CHECK THAT BUFFER FULL AND

```
 82 012234                                          BR      12$             ; BRANCH
    012234  000000  012673                         ^O00,12$
 83 012236  106201  000004          8$:     CMP     #4,R1           ; SEE IF HEADER COMPARE FAILURE
 84 012240                                          BNE     6$              ; IF NOT, BRANCH
    012240  050000  012403                         ^O50000,6$
 85 012242  104653  000046                         MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
 86 012244  104141                                  MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
 87 012245                                          ASSUME  S.PARM,0
 88 012245  102201  020000                         BIT     #DCYLS,R1       ; SEE IF ON DIAGNOSTIC CYLINDERS
 89 012247                                          BNE     9$              ; IF SO, BRANCH
    012247  050000  012265                         ^O50000,9$
 90 012251  102203  000400                         BIT     #REVEC,R3       ; SEE IF REVECTOR ENABLED
 91 012253                                          BNE     5$              ; IF SO, BRANCH AROUND
    012253  050000  012374                         ^O50000,5$
 92 012255  102203  000200                         BIT     #RBNBN,R3       ; SEE IF READING RCT OR RBN
 93 012257                                          BNE     9$              ; IF SO, REPORT ERROR
    012257  050000  012265                         ^O50000,9$
 94 012261  104207  015007          2$:     MOV     #REVCT,R0       ; REVECTOR NEXT MODULE
 95 012263                                          BR      19$             ; EXIT
    012263  000000  013012                         ^O00,19$
 96 012265                          9$:     HARDER  19              ; REPORT HEADER COMPARE ERROR
    012265  104200  001754  001601                                          MOV     #ER19,HRQ.04
    012270  104202  107663                                                  MOV     #19!ERHARD+4000.,R2
    012272  104020  001577                                                  MOV     R2,HRQ.02
    012274  104200  012274  001576                                          MOV     #.,HRQ.01
    012277  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
 97 012302                                  ERRORC  <S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
    012302  104640  000005  001602                                          MOV     S.LETR(R4),HRQ.05
    012305  104670  000003  001603                                          MOV     RW.LOW(R0),HRQ.06
    012310  104670  000004  001604                                          MOV     RW.HI(R0),HRQ.07
 98 012313                                  ERRORC  <R3,#RBNTXT,U.RBN(R5)>
    012313  104030  001605                                                  MOV     R3,HRQ.08
    012315  104200  005472  001606                                          MOV     #RBNTXT,HRQ.09
    012320  104650  000055  001607                                          MOV     U.RBN(R5),HRQ.10
 99 012323                                  ERRORC  <U.RBN+1(R5),RW.ANG(R0)>
    012323  104650  000056  001610                                          MOV     U.RBN+1(R5),HRQ.11
    012326  104670  000007  001611                                          MOV     RW.ANG(R0),HRQ.12
100 012331                                  ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
    012331  104670  000005  001612                                          MOV     RW.CMD(R0),HRQ.13
    012334  104650  000066  001613                                          MOV     U.CGRP(R5),HRQ.14
    012337  104650  000064  001614                                          MOV     U.CCYL(R5),HRQ.15
    012342  104650  000065  001615                                          MOV     U.CCYL+1(R5),HRQ.16
101 012345                                  ERRORC  <U.LGRP(R5),U.LCYL(R5),U.LCYL+1(R5)>
    012345  104650  000071  001616                                          MOV     U.LGRP(R5),HRQ.17
    012350  104650  000067  001617                                          MOV     U.LCYL(R5),HRQ.18
    012353  104650  000070  001620                                          MOV     U.LCYL+1(R5),HRQ.19
102 012356                                  ENDERR  0
    012356  114000  003013                                                  CLR     ERRPOS          ; CLEAR THE POSITION
103 012360  103203  000200                 BIC     #RBNBN,R3       ; IF HANDLING A RBN, CLEAR IT
104 012362  100653  000046                 MOV     R3,U.PARM(R5)   ; SAVE
105 012364  104201  000001                 MOV     #1,R1           ; ONLY ONE SECTOR HANDLED
106 012366  100651  000021                 MOV     R1,U.NSEC(R5)   ; SAVE
107 012370  104207  005134                 MOV     #SETUP,R0       ; SETUP NEXT MODULE
108 012372                                 BR      18$             ; BRANCH + REPORT, DELAYED CALL
    012372  000000  013014                 ^O00,18$
109 012374  104207  015007          5$:    MOV     #REVCT,R0       ; REVECTOR NEXT MODULE
110 012376  104200  177777  003003        MOV     #-1,SCR2        ; SECTOR MARKED BAD
```

```
    111 012401                                          BR      19$             ; EXIT, IMMIDATE CALL
        012401  000000  013012                         ^O0,19$
    112 012403  106201  000052          6$:     CMP     #52,R1          ; SEE IF SERDES OVERRUN ERROR
    113 012405                                          BNE     20$             ; IF NOT, BRANCH
        012405  050000  012426                         ^O50000,20$
    114 012407                                          SOFTER  34              ; SERDES OVERRUN
        012407  104200  002540  001601                                                  MOV     #ER34,HRQ.04
        012412  104202  147702                                                  MOV     #34!ERSOFT+4000.,R2
        012414  104020  001577                                                  MOV     R2,HRQ.02
        012416  104200  012416  001576                                         MOV     #.,HRQ.01
        012421  104200  060013  001575                                         MOV     #ERRMES,HRQ.RQ
    115 012424                                          BR      12$             ; REST OF ERROR MESSAGE
        012424  000000  012673                         ^O0,12$
    116 012426  106201  000150          20$:    CMP     #150,R1         ; SEE IF DATA SYNC TIMEOUT
    117 012430                                          BNE     22$             ; IF NOT, BRANCH
        012430  050000  012451                         ^O50000,22$
    118 012432                                          SOFTER  36              ; DATA SYNC TIMEOUT
        012432  104200  002612  001601                                                  MOV     #ER36,HRQ.04
        012435  104202  147704                                                  MOV     #36!ERSOFT+4000.,R2
        012437  104020  001577                                                  MOV     R2,HRQ.02
        012441  104200  012441  001576                                         MOV     #.,HRQ.01
        012444  104200  060013  001575                                         MOV     #ERRMES,HRQ.RQ
    119 012447                                          BR      12$             ; REST OF ERROR MESSAGE
        012447  000000  012673                         ^O0,12$
    120 012451  106201  000153          22$:    CMP     #153,R1         ; SEE IF POSITIONER ERROR
    121 012453                                          BNE     23$             ; IF NOT, BRANCH
        012453  050000  012521                         ^O50000,23$
    122 012455                                          REPSFT  .,SEEK          ; REPORT SEEK ERROR
        012455  114000  001577                                                  CLR     HRQ.02
        012457  114000  001600                                                  CLR     HRQ.03
        012461  104200  000001  001601                                         MOV     #1,HRQ.04
        012464  100467                                                                  MOV R0,-(SP)
        012465  104657  000063                                                  MOV     U.UNUM(R5),R0
        012467  105657  000050                                                  ADD     U.SUBU(R5),R0
        012471  104070  001576                                                  MOV     R0,HRQ.01
        012473  104207  060007                                                  MOV     #T4SOFT,R0
        012475  020000  001543                         ^O20000,HOSTRQ
        012477  104267                                                                  MOV (SP)+,R0
    123 012500                                          SOFTER  45              ; REPORT POSITIONER ERROR
        012500  104200  003162  001601                                                  MOV     #ER45,HRQ.04
        012503  104202  147715                                                  MOV     #45!ERSOFT+4000.,R2
        012505  104020  001577                                                  MOV     R2,HRQ.02
        012507  104200  012507  001576                                         MOV     #.,HRQ.01
        012512  104200  060013  001575                                         MOV     #ERRMES,HRQ.RQ
    124 012515                                          CALL    GORCLB          ; RECALIBRATION REQUIRED FOR RECOVERY
        012515  020000  005076                         ^O20000,GORCLB
    125 012517                                          BR      12$             ; REST OF ERROR MESSAGE
        012517  000000  012673                         ^O0,12$
    126 012521  106201  000213          23$:    CMP     #213,R1         ; SEE IF READ/WRITE READY FAILURE
    127 012523                                          BNE     24$             ; IF NOT, BRANCH
        012523  050000  012544                         ^O50000,24$
    128 012525                                          SOFTER  37              ; READ/WRITE READY FAILURE
        012525  104200  002634  001601                                                  MOV     #ER37,HRQ.04
        012530  104202  147705                                                  MOV     #37!ERSOFT+4000.,R2
        012532  104020  001577                                                  MOV     R2,HRQ.02
        012534  104200  012534  001576                                         MOV     #.,HRQ.01
        012537  104200  060013  001575                                         MOV     #ERRMES,HRQ.RQ
```

```
129 012542                                              BR      12$             ; REST OF ERROR MESSAGE
    012542  000000  012673                               ^O0,12$
130 012544  106201  000253          24$:                CMP     #253,R1         ; SEE IF DATA DRIVE OR STATE CLOCK TIMEOUT
131 012546                                              BNE     21$             ; IF NOT, BRANCH
    012546  050000  012571                               ^O50000,21$
132 012550                                              SOFTER  35              ; DRIVE CLOCK TIMEOUT
    012550  104200  002563  001601                                                      MOV     #ER35,HRQ.04
    012553  104202  147703                                                             MOV     #35!ERSOFT+4000.,R2
    012555  104020  001577                                                             MOV     R2,HRQ.02
    012557  104200  012557  001576                                                     MOV     #.,HRQ.01
    012562  104200  060013  001575                                                     MOV     #ERRMES,HRQ.RQ
133 012565                                              CALL    GOINIT          ; INIT THE DRIVE TO RECOVER
    012565  020000  005106                               ^O20000,GOINIT
134 012567                                              BR      12$             ; REST OF ERROR MESSAGE
    012567  000000  012673                               ^O0,12$
135 012571  106201  000313          21$:                CMP     #313,R1         ; SEE IF RECIEVER READY TIMEOUT
136 012573                                              BNE     25$             ; IF NOT, BRANCH
    012573  050000  012616                               ^O50000,25$
137 012575                                              SOFTER  38              ; RECIEVER READY FAILURE
    012575  104200  002660  001601                                                      MOV     #ER38,HRQ.04
    012600  104202  147706                                                             MOV     #38!ERSOFT+4000.,R2
    012602  104020  001577                                                             MOV     R2,HRQ.02
    012604  104200  012604  001576                                                     MOV     #.,HRQ.01
    012607  104200  060013  001575                                                     MOV     #ERRMES,HRQ.RQ
138 012612                                              CALL    GOINIT          ; INIT THE DRIVE TO RECOVER
    012612  020000  005106                               ^O20000,GOINIT
139 012614                                              BR      12$             ; REST OF ERROR MESSAGE
    012614  000000  012673                               ^O0,12$
140 012616  106201  000413          25$:                CMP     #413,R1         ; RTDS FAILURE DURING READ?
141 012620                                              BNE     26$             ; IF NOT, BRANCH
    012620  050000  012641                               ^O50000,26$
142 012622                                              SOFTER  64              ; REPORT
    012622  104200  004145  001601                                                      MOV     #ER64,HRQ.04
    012625  104202  147740                                                             MOV     #64!ERSOFT+4000.,R2
    012627  104020  001577                                                             MOV     R2,HRQ.02
    012631  104200  012631  001576                                                     MOV     #.,HRQ.01
    012634  104200  060013  001575                                                     MOV     #ERRMES,HRQ.RQ
143 012637                                              BR      12$             ; BRANCH
    012637  000000  012673                               ^O0,12$
144 012641                          26$:                HARDER  69,<#SER36,R1>  ; UNKNOWN ERROR NUMBER
    012641  104200  004220  001601                                                      MOV     #ER69,HRQ.04
    012644  104200  004242  001602                                                     MOV     #SER36,HRQ.05
    012647  104010  001603                                                             MOV     R1,HRQ.06
    012651  104202  107745                                                             MOV     #69!ERHARD+4000.,R2
    012653  104020  001577                                                             MOV     R2,HRQ.02
    012655  104200  012655  001576                                                     MOV     #.,HRQ.01
    012660  104200  060014  001575                                                     MOV     #ERRMC,HRQ.RQ
145 012663                                              ENDERR
    012663  104200  000051  001604                                                     MOV     #SER22,HRQ.07
    012666  104200  000010  003013                                                     MOV     #7+1,ERRPOS     ; SET THE POSITION
146 012671                                              BR      14$             ; REST OF ERROR MESSAGE
    012671  000000  012764                               ^O0,14$
147 012673                          12$:                CERROR  5,#SER19
    012673  104200  000470  001602                                                     MOV     #SER19,HRQ.05
148 012676                                              ERRORC  <U.RWTO(R5),S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
    012676  104650  000012  001603                                                     MOV     U.RWTO(R5),HRQ.06
    012701  104640  000005  001604                                                     MOV     S.LETR(R4),HRQ.07
```

********** OVERLAY MODULE SECCHK - CHECK THAT BUFFER FULL AND

```
        012704  104670  000003  001605                                                  MOV     RW.LOW(R0),HRQ.08
        012707  104670  000004  001606                                                  MOV     RW.HI(R0),HRQ.09
    149 012712                                  ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5)>
        012712  104650  000046  001607                                                  MOV     U.PARM(R5),HRQ.10
        012715  104200  005472  001610                                                  MOV     #RBNTXT,HRQ.11
        012720  104650  000055  001611                                                  MOV     U.RBN(R5),HRQ.12
    150 012723                                  ERRORC  <U.RBN+1(R5),RW.ANG(R0)>
        012723  104650  000056  001612                                                  MOV     U.RBN+1(R5),HRQ.13
        012726  104670  000007  001613                                                  MOV     RW.ANG(R0),HRQ.14
    151 012731                                  ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
        012731  104670  000005  001614                                                  MOV     RW.CMD(R0),HRQ.15
        012734  104650  000066  001615                                                  MOV     U.CGRP(R5),HRQ.16
        012737  104650  000064  001616                                                  MOV     U.CCYL(R5),HRQ.17
        012742  104650  000065  001617                                                  MOV     U.CCYL+1(R5),HRQ.18
    152 012745                                  ERRORC  <U.LGRP(R5),U.LCYL(R5),U.LCYL+1(R5)>
        012745  104650  000071  001620                                                  MOV     U.LGRP(R5),HRQ.19
        012750  104650  000067  001621                                                  MOV     U.LCYL(R5),HRQ.20
        012753  104650  000070  001622                                                  MOV     U.LCYL+1(R5),HRQ.21
    153 012756                                  ENDERR
        012756  104200  000051  001623                                                  MOV     #SER22,HRQ.22
        012761  104200  000027  003013                                                  MOV     #22+1,ERRPOS    ; SET THE POSITION
    154 012764                          14$:    CALL    GOSEEK          ; SEEK REQUIRED FOR RECOVERY
        012764  020000  005066                  ^O020000,GOSEEK
    155 012766  104207  000001                  MOV     #1,R0           ; SET UP TO READ ONE SECTOR
    156 012770  100657  000022                  MOV     R0,U.MSEC(R5)   ; SAVE
    157 012772                                  BR      18$             ; EXIT
        012772  000000  013014                  ^O00,18$
    158 012774  104641  000010          3$:     MOV     S.MEGR(R4),R1   ; GET NUMBER OF SECTORS READ
    159 012776  117401                          DEC     R1              ; READJUST TO ACTUAL SECTORS TESTED
    160 012777  100641  000010                  MOV     R1,S.MEGR(R4)   ; SAVE
    161 013001  104201  100000                  MOV     #EOC,R1         ; MOVE END-OF-CHAIN TO R1
    162 013003  100171                          MOV     R1,(R0)         ; FLAG THIS BUFFER AS END-OF-CHAIN (FOR LSTMOD)
    163 013004  104207  014726                  MOV     #LSTMOD,R0      ; LSTMOD NEXT MODULE
    164 013006                                  BR      19$             ; EXIT, NO ERRORS, IMMIDATE CALL
        013006  000000  013012                  ^O00,19$
    165 013010  104207  013016          4$:     MOV     #CHKEDC,R0      ; CHKEDC IS NEXT MODULE
    166 013012  114002                  19$:    CLR     R2              ; NO ERRORS
    167 013013  114001                  17$:    CLR     R1              ; IMMIDATE CALL
    168 013014                          18$:    BR      JMPRET          ; RETURN TO SEQUNCR
        013014  000000  003746                  ^O00,JMPRET
    169                                         .DSABL  LSB
```

```
                                .SBTTL  ********** OVERLAY MODULE CHKEDC - CHECK THAT ECC AND EDC ARE OK
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                ;
013016                          CHKEDC:
                                ;
                                ;       CHECK THAT THERE IS NO ECC ERROR IN THIS BUFFER, IF NOT, CHECK THE
                                ;       EDC, IF THAT IS INCORRECT YOU HAVE A MAJOR ERROR...
                                ;       IF ECC AND EDC IS OK, GO TO DATA COMPARE (IF REQUESTED)
                                ;
                                        .ENABL  LSB
013016  104307  003015                  MOV     CHAINS,R0       ; R0 POINTS AT READ CHAIN LINK BEING TESTED
013020  104671  000001                  MOV     RW.STAT(R0),R1  ; R1 IS BUFFER STATUS
013022  102201  010000                  BIT     #ECCFLG,R1      ; SEE IF ECC ERROR IN BUFFER
013024                                  BEQ     11$             ; IF NOT, CHECK EDC
013024  010000  013164                  ^010000,11$
013026  104653  000046                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
013030  102203  000400                  BIT     #REVEC,R3       ; SEE IF FINDING A REVECTOR
013032                                  BNE     2$              ; IF SO, BRANCH
013032  050000  013060                  ^050000,2$
013034                                  CALL    BLKCHK          ; SEE IF THIS IS A KNOWN BAD BLOCK
013034  020000  002465                  ^020000,BLKCHK
013036                                  BCS     5$              ; IF NOT, BRANCH
013036  040000  013042                  ^040000,..+3
013040  000000  013053                  ^00,5$
013042  104641  000010                  MOV     S.MEGR(R4),R1   ; GET NUMBER OF SECTORS READ
013044  117401                          DEC     R1              ; READJUST TO ACTUAL SECTORS HANDLED
013045  100641  000010                  MOV     R1,S.MEGR(R4)   ; SAVE
013047  104207  014726                  MOV     #LSTMOD,R0      ; LAST MODULE IS NEXT
013051                                  BR      19$             ; EXIT, NO ERRORS, IMMIDATE CALL
013051  000000  013433                  ^00,19$
013053  104141                  5$:     MOV     (R4),R1         ; GET UNIT PARAMETERS
013054                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
013054  102201  010000                  BIT     #ECCCHK,R1      ; SEE IF ECC CORRECTION IS REQUESTED
013056                                  BEQ     6$              ; IF NOT, BRANCH
013056  010000  013064                  ^010000,6$
013060  104207  013437          2$:     MOV     #CHKECC,R0      ; CHKECC NEXT MODULE
013062                                  BR      19$             ; EXIT, NO ERRORS, IMMIDATE CALL
013062  000000  013433                  ^00,19$
013064                          6$:     SOFTER  7,<#SER21,U.RRTY(R5),U.ELEV(R5)>
013064  104200  000615  001601                                          MOV     #ER7,HRQ.04
013067  104200  000661  001602                                          MOV     #SER21,HRQ.05
013072  104650  000010  001603                                          MOV     U.RRTY(R5),HRQ.06
013075  104650  000027  001604                                          MOV     U.ELEV(R5),HRQ.07
013100  104202  147647                                                  MOV     #7!ERSOFT+4000.,R2
013102  104020  001577                                                  MOV     R2,HRQ.02
013104  104200  013104  001576                                          MOV     #.,HRQ.01
013107  104200  060013  001575                                          MOV     #ERRMES,HRQ.RQ
013112                          ERRORC  <S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
013112  104640  000005  001605                                          MOV     S.LETR(R4),HRQ.08
013115  104670  000003  001606                                          MOV     RW.LOW(R0),HRQ.09
013120  104670  000004  001607                                          MOV     RW.HI(R0),HRQ.10
013123                          ERRORC  <U.PARM(R5),#RBNTXT>
```

```
                013123  104650  000046  001610                                                  MOV     U.PARM(R5),HRQ.11
                013126  104200  005472  001611                                                  MOV     #RBNTXT,HRQ.12
     76 013131                                          ERRORC  <U.RBN(R5),U.RBN+1(R5),RW.ANG(R0)>
                013131  104650  000055  001612                                                  MOV     U.RBN(R5),HRQ.13
                013134  104650  000056  001613                                                  MOV     U.RBN+1(R5),HRQ.14
                013137  104670  000007  001614                                                  MOV     RW.ANG(R0),HRQ.15
     77 013142                                          ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5)>
                013142  104670  000005  001615                                                  MOV     RW.CMD(R0),HRQ.16
                013145  104650  000066  001616                                                  MOV     U.CGRP(R5),HRQ.17
                013150  104650  000064  001617                                                  MOV     U.CCYL(R5),HRQ.18
     78 013153                                          ERRORC  U.CCYL+1(R5)
                013153  104650  000065  001620                                                  MOV     U.CCYL+1(R5),HRQ.19
     79 013156                                          ENDERR  0
                013156  114000  003013                                                          CLR     ERRPOS          ; CLEAR THE POSITION
     80 013160  104651  000046                          MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
     81 013162                                          BR      10$             ; EXIT
                013162  000000  013371                  ^O0,10$
     82 013164  104677  000002              11$:        MOV     RW.BUF(R0),R0   ; R0 NOW POINTS AT BUFFER
     83 013166                                          CALL    CMPEDC          ; COMPUTE THE EDC
                013166  020000  001640                  ^O020000,CMPEDC
     84 013170  104651  000046                          MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
     85 013172  106672  000400                          CMP     BF.EDC(R0),R2   ; SEE IF MATCH
     86 013174                                          BNE     14$             ; IF NOT, BRANCH
                013174  050000  013261                  ^O050000,14$
     87 013176  104657  000010                          MOV     U.RRTY(R5),R0   ; GET RETRIES
     88 013200                                          BNE     23$             ; IF RETRIES WERE USED, REPORT SOFT ERROR
                013200  050000  013210                  ^O050000,23$
     89 013202  104657  000027                          MOV     U.ELEV(R5),R0   ; GET ERROR LEVEL
     90 013204  106657  000031                          CMP     U.MLEV(R5),R0   ; SEE IF ANY ERROR LEVELS WERE USED
     91 013206                                          BEQ     22$             ; IF NOT, BRANCH
                013206  010000  013233                  ^O010000,22$
     92 013210                          23$:            REPSFT  SOFT            ; REPORT SOFT ERROR
                013210  104200  000001  001577                                                  MOV     #1,HRQ.02
                013213  114000  001600                                                          CLR     HRQ.03
                013215  114000  001601                                                          CLR     HRQ.04
                013217  100467                                                                          MOV R0,-(SP)
                013220  104657  000063                                                          MOV     U.UNUM(R5),R0
                013222  105657  000050                                                          ADD     U.SUBU(R5),R0
                013224  104070  001576                                                          MOV     R0,HRQ.01
                013226  104207  060007                                                          MOV     #T4SOFT,R0
                013230  020000  001543                  ^O020000,HOSTRQ
                013232  104267                                                                          MOV (SP)+,R0
     93 013233  114000  003003              22$:        CLR     SCR2            ; FLAG AS GOOD BLOCK
     94 013235  102201  000400                          BIT     #REVEC,R1       ; SEE IF REVECTOR IN PROGRESS
     95 013237                                          BEQ     1$              ; IF NOT, BRANCH
                013237  010000  013245                  ^O010000,1$
     96 013241  104207  015007                          MOV     #REVCT,R0       ; REVECTOR NEXT MODULE
     97 013243                                          BR      19$             ; EXIT
                013243  000000  013433                  ^O0,19$
     98 013245  102201  000002              1$:         BIT     #DATCMP,R1      ; SEE IF DATA COMPARE REQUESTED
     99 013247                                          BNE     12$             ; IF SO, BRANCH
                013247  050000  013255                  ^O050000,12$
    100 013251  104207  014726                          MOV     #LSTMOD,R0      ; LSTMOD NEXT MODULE
    101 013253                                          BR      19$             ; EXIT, NO ERRORS, IMMIDATE CALL
                013253  000000  013433                  ^O0,19$
    102 013255  104207  014403              12$.        MOV     #CMPDAT,R0      ; DATA COMPARE NEXT MODULE
    103 013257                                          BR      19$             ; EXIT, NO ERRORS, IMMIDATE CALL
```

```
                013257  000000  013433                  ^O00,19$
    104 013261  104307  003015          14$:    MOV     CHAINS,R0
    105 013263  104020  003003                  MOV     R2,SCR2         ; SAVE CALCULATED EDC
    106 013265  104673  000002                  MOV     RW.BUF(R0),R3
    107 013267                                  SOFTER  24,<#SER21,U.RRTY(R5),U.ELEV(R5)>
        013267  104200  002272  001601                                    MOV     #ER24,HRQ.04
        013272  104200  000661  001602                                    MOV     #SER21,HRQ.05
        013275  104650  000010  001603                                    MOV     U.RRTY(R5),HRQ.06
        013300  104650  000027  001604                                    MOV     U.ELEV(R5),HRQ.07
        013303  104202  147670                                           MOV     #24!ERSOFT+4000.,R2
        013305  104020  001577                                           MOV     R2,HRQ.02
        013307  104200  013307  001576                                    MOV     #.,HRQ.01
        013312  104200  060013  001575                                    MOV     #ERRMES,HRQ.RQ
    108 013315                                  ERRORC  <S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
        013315  104640  000005  001605                                    MOV     S.LETR(R4),HRQ.08
        013320  104670  000003  001606                                    MOV     RW.LOW(R0),HRQ.09
        013323  104670  000004  001607                                    MOV     RW.HI(R0),HRQ.10
    109 013326                                  ERRORC  <U.PARM(R5),#RBNTXT>
        013326  104650  000046  001610                                    MOV     U.PARM(R5),HRQ.11
        013331  104200  005472  001611                                    MOV     #RBNTXT,HRQ.12
    110 013334                                  ERRORC  <U.RBN(R5),U.RBN+1(R5),RW.ANG(R0)>
        013334  104650  000055  001612                                    MOV     U.RBN(R5),HRQ.13
        013337  104650  000056  001613                                    MOV     U.RBN+1(R5),HRQ.14
        013342  104670  000007  001614                                    MOV     RW.ANG(R0),HRQ.15
    111 013345                                  ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5)>
        013345  104670  000005  001615                                    MOV     RW.CMD(R0),HRQ.16
        013350  104650  000066  001616                                    MOV     U.CGRP(R5),HRQ.17
        013353  104650  000064  001617                                    MOV     U.CCYL(R5),HRQ.18
    112 013356                                  ERRORC  <U.CCYL+1(R5),SCR2,BF.EDC(R3)>
        013356  104650  000065  001620                                    MOV     U.CCYL+1(R5),HRQ.19
        013361  104300  003003  001621                                    MOV     SCR2,HRQ.20
        013364  104630  000400  001622                                    MOV     BF.EDC(R3),HRQ.21
    113 013367                                  ENDERR  0
        013367  114000  003013                                           CLR     ERRPOS          ; CLEAR THE POSITION
    114 013371  104200  177777  003003  10$:    MOV     #-1,SCR2        ; FLAG AS BLOCK BAD
    115 013374  102201  000400                  BIT     #REVEC,R1       ; SEE IF REVECTOR IN PROGRESS
    116 013376                                  BNE     13$             ; IF SO, RETRY
        013376  050000  013405                  ^O50000,13$
    117 013400  104147                  15$:    MOV     (R4),R0         ; GET SUBUNIT PARAMETERS
    118 013401                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
    119 013401  102207  001000                  BIT     #RTRIES,R0      ; SEE IF RETRIES ARE ENABLED
    120 013403                                  BEQ     16$             ; IF NOT, BRANCH
        013403  010000  013411                  ^O10000,16$
    121 013405  104207  014104          13$:    MOV     #ERCOV,R0       ; ERROR RECOVERY IS NEXT MODULE
    122 013407                                  BR      20$             ; EXIT, IMMIDATE CALL
        013407  000000  01343'                  ^O0,20$
    123 013411  103200  040000  001577  16$:    BIC     #C2HARD,HRQ.02  ; MAKE A HARD ERROR
    124 013414  104200  060014  001575          MOV     #ERRMC,HRQ.RQ   ; COUNT ERROR
    125 013417  102201  000002                  BIT     #DATCMP,R1      ; SEE IF DATA COMPARE ENABLED
    126 013421                                  BEQ     17$             ; IF NOT, BRANCH
        013421  010000  013427                  ^O10000,17$
    127 013423  10420?  014403                  MOV     #CMPDAT,R0      ; DATA COMPARE IS NEXT MODULE
    128 013425                                  BR      20$             ; EXIT, IMMIDATE CALL
        013425  000000  013434                  ^O0,20$
    129 013427  104207  014726          17$:    MOV     #LSTMOD,R0      ; LSTMOD IS NEXT MODULE
    130 013431                                  BR      20$             ; EXIT, IMMIDATE CALL
        013431  000000  013434                  ^O0,20$
```

```
    131 013433  114002                    19$:    CLR     R2              ; NO ERRORS
    132 013434  114001                    20$:    CLR     R1              ; IMMIDATE CALL
    133 013435                                    BR      JMPRET          ; RETURN TO SEQUNCR
        013435  000000  003746                    ^O0,JMPRET
    134                                           .DSABL  LSB
```

```
                                .SBTTL  ********** OVERLAY MODULE CHKECC - CORRECT BUFFER READ USING ECC
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                ;
013437                          CHKECC:
                                ;
                                ;       SEE IF ECC ERROR WAS DECTECTED
                                ;
                                        .ENABL  LSB
013437  104307  003015                  MOV     CHAINS,R0       ; R0 POINTS TO CHAIN
013441                                  PUSH    R0              ; SAVE POINTER TO LINK
013441  100467                                                                  MOV R0,-(SP)
013442  104200  177777  003003          MOV     #-1,SCR2        ; ASSUME BUFFER BAD
013445  104202  000400                  MOV     #SEC512,R2      ; CORRECT ECC ON A 512 BYTE SECTOR
013447  060015                          XFC     ECC             ; APPLY ECC CORRECTION
013450  115001                          TST     R1              ; SEE IF CORRECTION WORKED
013451                                  BEQ     1$              ; IF SO, BRANCH
013451  010000  013473                  ^010000,1$
013453                                  POP     R0              ; RESTORE POINTER TO LINK
013453  104267                                                                  MOV (SP)+,R0
013454                                  SOFTER  8
013454  104200  000631  001601                                                  MOV     #ER8,HRQ.04
013457  104202  147650                                                          MOV     #8!ERSOFT+4000.,R2
013461  104020  001577                                                          MOV     R2,HRQ.02
013463  104200  013463  001576                                                  MOV     #.,HRQ.01
013466  104200  060013  001575                                                  MOV     #ERRMES,HRQ.RQ
013471                                  BR      8$              ; BRANCH
013471  000000  013517                  ^00,8$
013473  106657  000032          1$:     CMP     U.ECCT(R5),R0   ; SEE IF CORRECTIONS = TO OR EXCEED THRESHOLD
013475                                  BEQ     6$              ; IF =, BRANCH
013475  010000  013501                  ^010000,6$
013477                                  BPL     2$              ; IF NOT, BRANCH
013477  030000  013600                  ^030000,2$
013501                          6$:     POP     R0              ; RESTORE POINTER TO LINK
013501  104267                                                                  MOV (SP)+,R0
013502                                  SOFTER  9
013502  104200  000745  001601                                                  MOV     #ER9,HRQ.04
013505  104202  147651                                                          MOV     #9!ERSOFT+4000.,R2
013507  104020  001577                                                          MOV     R2,HRQ.02
013511  104200  013511  001576                                                  MOV     #.,HRQ.01
013514  104200  060013  001575                                                  MOV     #ERRMES,HRQ.RQ
013517                          8$:     ERRORC  <#SER21,U.RRTY(R5),U.ELEV(R5)>
013517  104200  000661  001602                                                  MOV     #SER21,HRQ.05
013522  104650  000010  001603                                                  MOV     U.RRTY(R5),HRQ.06
013525  104650  000027  001604                                                  MOV     U.ELEV(R5),HRQ.07
013530                                  ERRORC  <S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
013530  104640  000005  001605                                                  MOV     S.LETR(R4),HRQ.08
013533  104670  000003  001606                                                  MOV     RW.LOW(R0),HRQ.09
013536  104670  000004  001607                                                  MOV     RW.HI(R0),HRQ.10
013541                                  ERRORC  <U.PARM(R5),#RBNTXT>
013541  104650  000046  001610                                                  MOV     U.PARM(R5),HRQ.11
013544  104200  005472  001611                                                  MOV     #RBNTXT,HRQ.12
013547                                  ERRORC  <U.RBN(R5),U.RBN+1(R5),RW.ANG(R0)>
```

```
                013547  104650  000055  001612                                                  MOV     U.RBN(R5),HRQ.13
                013552  104650  000056  001613                                                  MOV     U.RBN+1(R5),HRQ.14
                013555  104670  000007  001614                                                  MOV     RW.ANG(R0),HRQ.15
     64 013560                                  ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5)>
                013560  104670  000005  001615                                                  MOV     RW.CMD(R0),HRQ.16
                013563  104650  000066  001616                                                  MOV     U.CGRP(R5),HRQ.17
                013566  104650  000064  001617                                                  MOV     U.CCYL(R5),HRQ.18
     65 013571                                  ERRORC  U.CCYL+1(R5)
                013571  104650  000065  001620                                                  MOV     U.CCYL+1(R5),HRQ.19
     66 013574                                  ENDERR  0
                013574  114000  003013                                                          CLR     ERRPOS          ; CLEAR THE POSITION
     67 013576                                  BR      11$             ; BRANCH
                013576  000000  014025                                  ^O0,11$
     68 013600                          2$:     POP     R3              ; RESTORE POINTER TO BUFFER
                013600  104263                                                                  MOV (SP)+,R3
     69 013601  104637  000002                  MOV     RW.BUF(R3),R0   ; R0 POINTS TO BUFFER
     70 013603                                  CALL    CMPEDC          ; COMPUTE EDC VALUE
                013603  020000  001640                                  ^O20000,CMPEDC
     71 013605  106672  000400                  CMP     BF.EDC(R0),R2   ; SEE IF EDC VALUE MATCHES
     72 013607                                  BNE     3$              ; IF NOT, BRANCH
                013607  050000  013721                                  ^O50000,3$
     73 013611  104200  000001  001600          MOV     #1,HRQ.03       ; REPORT ECC CORRECTION
     74 013614  104657  000010                  MOV     U.RRTY(R5),R0   ; GET TIMEOUT
     75 013616                                  BNE     23$             ; IF RETRIES WERE USED, REPORT SOFT ERROR
                013616  050000  013630                                  ^O50000,23$
     76 013620  104657  000027                  MOV     U.ELEV(R5),R0   ; GET ERROR LEVEL
     77 013622  114000  001577                  CLR     HRQ.02          ; ASSUME NO SOFT ERROR
     78 013624  106657  000031                  CMP     U.MLEV(R5),R0   ; SEE IF ANY ERROR LEVELS WERE USED
     79 013626                                  BEQ     22$             ; IF NOT, BRANCH
                013626  010000  013632                                  ^O10000,22$
     80 013630  115400  001577          23$:    INC     HRQ.02          ; REPORT SOFT ERROR
     81 013632                          22$:    PUSH    R0              ; SAVE R0
                013632  100467                                                                  MOV R0,-(SP)
     82 013633  104657  000063                  MOV     U.UNUM(R5),R0   ; GET STARTING SUBUNIT NUMBER
     83 013635  105657  000050                  ADD     U.SUBU(R5),R0   ; ADD OFFSET
     84 013637  104070  001576                  MOV     R0,HRQ.01       ; MOVE TO OUTPUT BUFFER
     85 013641  104207  060007                  MOV     #T4SOFT,R0      ; REPORT REQUEST
     86 013643                                  CALL    HOSTRQ          ; SEND TO HOST
                013643  020000  001543                                  ^O20000,HOSTRQ
     87 013645                                  POP     R0              ; RESTORE R0
                013645  104267                                                                  MOV (SP)+,R0
     88 013646  114000  003003                  CLR     SCR2            ; BUFFER GOOD
     89 013650                                  MSSG    3,<S.LETR(R4),RW.LOW(R3),RW.HI(R3),RW.ANG(R3),RW.CMD(R3),U.CGRP(R5),U.CCYL(R
                013650  104200  005616  001577                                                  MOV     #MS3,HRQ.02
                013653  104640  000005  001600                                                  MOV     S.LETR(R4),HRQ.03
                013656  104630  000003  001601                                                  MOV     RW.LOW(R3),HRQ.04
                013661  104630  000004  001602                                                  MOV     RW.HI(R3),HRQ.05
                013664  104630  000007  001603                                                  MOV     RW.ANG(R3),HRQ.06
                013667  104630  000005  001604                                                  MOV     RW.CMD(R3),HRQ.07
                013672  104650  000066  001605                                                  MOV     U.CGRP(R5),HRQ.08
                013675  104650  000064  001606                                                  MOV     U.CCYL(R5),HRQ.09
                013700  104650  000065  001607                                                  MOV     U.CCYL+1(R5),HRQ.10
                013703  100467                                                                          MOV R0,-(SP)
                013704  10465C  000063  001576                                                  MOV     U.UNUM(R5),HRQ.01
                013707  105650  000050  001576                                                  ADD     U.SUBU(R5),HRQ.01
                013712  104207  060015                                                          MOV     #MESSAG,R0
                013714  020000  001543                                  ^O20000,HOSTRQ
```

```
                013716  104267                                                                                  MOV (SP)+,R0
     90 013717
                013717  000000  014054                BR      5$              ; EXIT
                                                      ^O00,5$
     91 013721  104020  003013          3$:    MOV     R2,ERRPOS       ; SAVE THE COMPUTED EDC
     92 013723                          SOFTER  10,<#SER21,U.RRTY(R5),U.ELEV(R5)>
                013723  104200  000770  001601                                    MOV     #ER10,HRQ.04
                013726  104200  000661  001602                                    MOV     #SER21,HRQ.05
                013731  104650  000010  001603                                    MOV     U.RRTY(R5),HRQ.06
                013734  104650  000027  001604                                    MOV     U.ELEV(R5),HRQ.07
                013737  104202  147652                                            MOV     #10!ERSOFT+4000.,R2
                013741  104020  001577                                            MOV     R2,HRQ.02
                013743  104200  013743  001576                                    MOV     #.,HRQ.01
                013746  104200  060013  001575                                    MOV     #ERRMES,HRQ.RQ
     93 013751                          ERRORC  <S.LETR(R4),RW.LOW(R3),RW.HI(R3)>
                013751  104640  000005  001605                                    MOV     S.LETR(R4),HRQ.08
                013754  104630  000003  001606                                    MOV     RW.LOW(R3),HRQ.09
                013757  104630  000004  001607                                    MOV     RW.HI(R3),HRQ.10
     94 013762                          ERRORC  <U.PARM(R5),#RBNTXT>
                013762  104650  000046  001610                                    MOV     U.PARM(R5),HRQ.11
                013765  104200  005472  001611                                    MOV     #RBNTXT,HRQ.12
     95 013770                          ERRORC  <U.RBN(R5),U.RBN+1(R5),RW.ANG(R3)>
                013770  104650  000055  001612                                    MOV     U.RBN(R5),HRQ.13
                013773  104650  000056  001613                                    MOV     U.RBN+1(R5),HRQ.14
                013776  104630  000007  001614                                    MOV     RW.ANG(R3),HRQ.15
     96 014001                          ERRORC  <RW.CMD(R3),U.CGRP(R5),U.CCYL(R5)>
                014001  104630  000005  001615                                    MOV     RW.CMD(R3),HRQ.16
                014004  104650  000066  001616                                    MOV     U.CGRP(R5),HRQ.17
                014007  104650  000064  001617                                    MOV     U.CCYL(R5),HRQ.18
     97 014012                          ERRORC  <U.CCYL+1(R5),ERRPOS,BF.EDC(R0)>        ; REPORT ERROR
                014012  104650  000065  001620                                    MOV     U.CCYL+1(R5),HRQ.19
                014015  104300  003013  001621                                    MOV     ERRPOS,HRQ.20
                014020  104670  000400  001622                                    MOV     BF.EDC(R0),HRQ.21
     98 014023                          ENDERR  0
                014023  114000  003013                                            CLR     ERRPOS          ; CLEAR THE POSITION
     99 014025  104143          11$:    MOV     (R4),R3         ; GET SUBUNIT PARAMETERS
    100 014026                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
    101 014026  102203  001000          BIT     #RTRIES,R3      ; SEE IF RETRIES ARE ENABLED
    102 014030                          BNE     4$              ; IF SO, BRANCH
                014030  050000  014050                          ^O50000,4$
    103 014032  104653  000046          MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
    104 014034  102203  000400          BIT     #REVEC,R3       ; SEE IF REVECTOR IN PROGRESS
    105 014036                          BNE     4$              ; IF SO, BRANCH
                014036  050000  014050                          ^O50000,4$
    106 014040  103200  040000  001577  BIC     #C2HARD,HRQ.02  ; MAKE SOFT ERROR A HARD ERROR
    107 014043  104200  060014  001575  MOV     #ERRMC,HRQ.RQ   ; COUNT ERROR
    108 014046                          BR      10$             ; BRANCH
                014046  000000  014067                          ^O00,10$
    109 014050  104207  014104    4$:   MOV     #ERCOV,R0       ; ERROR RECOVERY IS NEXT MODULE
    110 014052                          BR      7$              ; BRANCH
                014052  000000  014101                          ^O00,7$
    111 014054  114002          5$:     CLR     R2              ; NO ERRORS
    112 014055  104653  000046          MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
    113 014057  102203  000400          BIT     #REVEC,R3       ; SEE IF FINDING A REVECTOR
    114 014061                          BEQ     10$             ; IF NOT, BRANCH
                014061  010000  014067                          ^O10000,10$
    115 014063  104207  015007          MOV     #REVCT,R0       ; REVECTOR NEXT MODULE
    116 014065                          BR      7$              ; BRANCH
```

```
     014065  000000  014101                    ^O0,7$
117  014067  102203  000002          10$:      BIT     #DATCMP,R3      ; SEE IF DATA COMPARE IS REQUESTED
118  014071                                    BEQ     9$              ; IF NOT, BRANCH
     014071  010000  014077                    ^O10000,9$
119  014073  104207  014403                    MOV     #CMPDAT,R0      ; DATA COMPARE IS NEXT MODULE
120  014075                                    BR      7$              ; BRANCH
     014075  000000  014101                    ^O0,7$
121  014077  104207  014726          9$:       MOV     #LSTMOD,R0      ; GO TO LAST MODULE
122  014101  114001                  7$:       CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
123  014102                                    BR      JMPRET          ; RETURN
     014102  000000  003746                    ^O0,JMPRET
124                                            .DSABL  LSB
```

```
                                .SBTTL  ********** OVERLAY MODULE ERCOV - DATA ERROR RETRIES AND LEVELS
                                ;*************************************************************************
                                ;*************************************************************************
                                ;*************************************************************************
                                ;*************************************************************************
                                ;
                                ;
014104                          ERCOV:
                                ;
                                ;       ERROR RECOVERY AND RETRIES
                                ;
                                        .ENABL  LSB
014104                                  CALL    ENABLE          ; ENABLE ERROR RECOVERY
014104  020000  005036                  ^020000,ENABLE
014106  104657  000010          MOV     U.RRTY(R5),R0   ; GET NUMBER OF RETRIES ALLREADY ATTEMPTED
014110                          BNE     1$              ; IF NOT ZERO (FIRST TIME) BRANCH
014110  050000  014151                  ^050000,1$
014112  104651  000021          MOV     U.NSEC(R5),R1   ; GET NUMBER OF SECTORS READ
014114  105651  000024          ADD     U.CSEC(R5),R1   ; ADD NUMBER OF SECTORS PREVIOUSLY HANDLED
014116  100651  000024          MOV     R1,U.CSEC(R5)   ; SAVE
014120  104651  000021          MOV     U.NSEC(R5),R1   ; GET NUMBER OF SECTORS READ
014122  105641  000010          ADD     S.MEGR(R4),R1   ; ADD TO MEGABITS WRITTEN
014124  100641  000010          MOV     R1,S.MEGR(R4)   ; SAVE
014126  104651  000021          MOV     U.NSEC(R5),R1   ; GET NUMBER OF SECTORS READ
014130  114003                  CLR     R3              ; TO SETUP VALUES
014131  100653  000021          MOV     R3,U.NSEC(R5)   ; NO SECTORS READ
014133  115403                  INC     R3              ; MAKE R3 ONE
014134  100653  000022          MOV     R3,U.MSEC(R5)   ; ONLY READ ONE SECTOR
014136  105651  000053          ADD     U.CBN(R5),R1    ; ADJUST U.CBN TO SECTOR WITH ERROR
014140  100651  000053          MOV     R1,U.CBN(R5)    ; SAVE
014142                          BCC     1$              ; IF NO CARRY, BRANCH
014142  040000  014151                  ^040000,1$
014144  104651  000054          MOV     U.CBN+1(R5),R1  ; GET CURRENT BN
014146  115401                  INC     R1              ; PROPOGATE CARRY
014147  100651  000054          MOV     R1,U.CBN+1(R5)  ; SAVE
014151  106657  000030  1$:     CMP     U.RTRY(R5),R0   ; COMPARE MAXIMUM COUNT WITH RETRIES ATTEMPTED
014153                          BEQ     NLEV            ; IF RETRIES EXHAUSTED, BRANCH
014153  010000  014171                  ^010000,NLEV
014155  115407                  INC     R0              ; INCREMENT RETRY COUNT
014156  100657  000010          MOV     R0,U.RRTY(R5)   ; SAVE TIMEOUT
014160  104207  177777          MOV     #-1,R0          ; START READ RETRIES AT ZERO
014162  100657  000012          MOV     R0,U.RWTO(R5)   ; SAVE
014164  104207  007746          MOV     #BUILDP,R0      ; BUILDP IS NEXT MODULE
014166  104051                  MOV     R5,R1           ; DELAYED CALL TO READ
014167                          BR      EREXT           ; BRANCH
014167  000000  014202                  ^00,EREXT
014171  104207  014205  NLEV:  MOV     #NEWLEV,R0      ; NEW LEVEL OF ERROR RECOVERY WILL BE TRIED
014173  104653  000047          MOV     U.RCOV(R5),R3   ; GET ERROR RECOVERY PARAMETERS
014175  101203  010000          BIS     #NXTLEV,R3      ; GO TO NEXT LEVEL
014177  100653  000047          MOV     R3,U.RCOV(R5)   ; SAVE
014201  114001                  CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
014202  114002  EREXT:  CLR     R2              ; NO ERRORS
014203                          BR      JMPRET          ; RETURN TO SEQNCR
014203  000000  003746                  ^00,JMPRET
                                        .DSABL  LSB
```

```
                                .SBTTL  ********** OVERLAY MODULE NEWLEV - SEND DRIVE ERROR LEVEL
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                ;
014205                          NEWLEV:
                                ;
                                ;       INITIATE A NEW LEVEL OF ERROR RECOVERY
                                ;
                                        .ENABL  LSB
014205  104653  000047                  MOV     U.RCOV(R5),R3   ; GET RECOVERY WORD
014207  102203  010000                  BIT     #NXTLEV,R3      ; GO TO NEXT LEVEL?
014211                                  BNE     3$              ; IF SO, BRANCH
014211  050000  014220                  ^O50000,3$
014213  104657  000027                  MOV     U.ELEV(R5),R0   ; GET CURRENT ERROR RECOVERY LEVEL
014215  115407                          INC     R0              ; GO UP ONE LEVEL
014216                                  BR      LEVNZR
014216  000000  014334                  ^O00,LEVNZR
014220  104657  000027          3$:     MOV     U.ELEV(R5),R0   ; GET CURRENT ERROR RECOVERY LEVEL
014222                                  BNE     LEVNZR          ; IF NON-ZERO, BRANCH
014222  050000  014334                  ^O50000,LEVNZR
014224                                  CALL    DSABLE          ; DISABLE ERROR RECOVERY
014224  020000  005046                  ^O20000,DSABLE
014226                                  HARDER  11,<S.LETR(R4),U.CBN(R5),U.CBN+1(R5)>
014226  104200  001044  001601                                          MOV     #ER11,HRQ.04
014231  104640  000005  001602                                          MOV     S.LETR(R4),HRQ.05
014234  104650  000053  001603                                          MOV     U.CBN(R5),HRQ.06
014237  104650  000054  001604                                          MOV     U.CBN+1(R5),HRQ.07
014242  104202  107653                                                  MOV     #11!ERHARD+4000.,R2
014244  104020  001577                                                  MOV     R2,HRQ.02
014246  104200  014246  001576                                          MOV     #.,HRQ.01
014251  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
014254                                  ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5),U.RBN+1(R5),U.CGRP(R5)>
014254  104650  000046  001605                                          MOV     U.PARM(R5),HRQ.08
014257  104200  005472  001606                                          MOV     #RBNTXT,HRQ.09
014262  104650  000055  001607                                          MOV     U.RBN(R5),HRQ.10
014265  104650  000056  001610                                          MOV     U.RBN+1(R5),HRQ.11
014270  104650  000066  001611                                          MOV     U.CGRP(R5),HRQ.12
014273                                  ERRORC  <U.CCYL(R5),U.CCYL+1(R5)>
014273  104650  000064  001612                                          MOV     U.CCYL(R5),HRQ.13
014276  104650  000065  001613                                          MOV     U.CCYL+1(R5),HRQ.14
014301                                  ENDERR  0
014301  114000  003013                                                  CLR     ERRPOS          ; CLEAR THE POSITION
014303  104653  000046                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
014305  104200  177777  003003          MOV     #-1,SCR2        ; FLAG AS BAD SECTOR
014310  102203  000400                  BIT     #REVEC,R3       ; SEE IF FINDING A REVECTORED SECTOR
014312                                  BEQ     1$              ; IF NOT, BRANCH
014312  010000  014320                  ^O10000,1$
014314  104207  015007                  MOV     #REVCT,R0       ; GO TO REVCT ROUTINE NEXT
014316                                  BR      NEWEXT          ; BRANCH TO EXIT
014316  000000  014400                  ^O00,NEWEXT
014320  102203  000002          1$:     BIT     #DATCMP,R3      ; SEE IF DATA COMPARE REQUESTED
014322                                  BEQ     2$              ; IF NOT, BRANCH
014322  010000  014330                  ^O10000,2$
```

```
********** OVERLAY MODULE NEWLEV - SEND DRIVE ERROR LEVEL

     014324  104207  014403            MOV     #CMPDAT,R0      ; COMPARE DATA NEXT MODULE
     014326                            BR      NEWEXT          ; EXIT
     014326  000000  014400            ^O0,NEWEXT
     014330  104207  014726    2$:     MOV     #LSTMOD,R0      ; LAST MODULE NEXT MODULE
     014332                            BR      NEWEXT          ; EXIT
     014332  000000  014400            ^O0,NEWEXT
                                       .DSABL  LSB
     014334  104070  002437    LEVNZR: MOV     R0,ERRLEV       ; MOVE LEVEL TO SDI COMMAND
     014336  104203  002403            MOV     #CR.ERR,R3      ; POINT TO ERROR RECOVERY COMMAND
     014340                            CALL    TALK            ; SEND SDI INTERCHANGE
     014340  020000  001670            ^O20000,TALK
     014342  115002                    TST     R2              ; SEE IF AN ERROR OCCURRED
     014343                            BEQ     1$              ; IF NOT, BRANCH
     014343  010000  014352            ^O10000,1$
     014345                            CERROR  5,#SER4         ; REPORT ERROR
     014345  104200  004726  001602                                    MOV     #SER4,HRQ.05
     014350                            BR      SNDEX           ; BRANCH
     014350  000000  014401            ^O0,SNDEX
     014352  104657  000047    1$:     MOV     U.RCOV(R5),R0   ; GET RECOVERY PARAMETERS
     014354  102207  010000            BIT     #NXTLEV,R0      ; DID THIS GO TO THE NEXT LEVEL?
     014356                            BEQ     2$              ; IF NOT, BRANCH
     014356  010000  014373            ^O10000,2$
     014360  103207  010000            BIC     #NXTLEV,R0      ; CLEAR NEXT LEVEL BIT
     014362  101207  020000            BIS     #LEVUSD,R0      ; FLAG AS LEVELS USED
     014364  100657  000047            MOV     R0,U.RCOV(R5)   ; SAVE
     014366  104657  000027            MOV     U.ELEV(R5),R0   ; GET ERROR RECOVERY LEVEL
     014370  117407                    DEC     R0              ; DECREMENT
     014371  100657  000027            MOV     R0,U.ELEV(R5)   ; SAVE
     014373  104207  007746    2$:     MOV     #BUILDP,R0      ; BUILDP IS NEXT MODULE CALLED
     014375  114002                    CLR     R2              ; NO ERRORS
     014376  100652  000010            MOV     R2,U.RRTY(R5)   ; START RETRIES AT 0
     014400  114001            NEWEXT: CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
     014401                    SNDEX:  BR      JMPRET          ; RETURN TO SEQUENCER
     014401  000000  003746            ^O0,JMPRET
```

```
                                        .SBTTL  ********** OVERLAY MODULE CMPDAT - DATA COMPARISON ON READ BUFFER(S)
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;
                                        ;
                                        ;
014403                                  CMPDAT:
                                        ;
                                        ;       CMPDAT COMPARES THE PATTERN WITH THAT READ FROM THE SECTOR
                                        ;
                                                .ENABL  LSB
014403                                  PUSH    <R4,R5>         ; SAVE R4,R5
014403  100464                                                                  MOV R4,-(SP)
014404  100465                                                                  MOV R5,-(SP)
014405  104307  003015                  MOV     CHAINS,R0       ; R0 POINTS TO LINK
014407  104677  000002                  MOV     RW.BUF(R0),R0   ; R0 POINTS TO BUFFER
014411  104172                          MOV     (R0),R2         ; R2 HAS PATTERN NUMBER (IN EACH NIBBLE)
014412  103202  177760                  BIC     #LBLONB,R2      ; CLEAR UNUSED BITS
014414                                  BNE     1$              ; IF NOT PATTERN 16, BRANCH
014414  050000  014423                  ^O50000,1$
014416  104200  000020  003004          MOV     #16.,PNUM       ; REPORT PATTERN 16
014421                                  BR      2$              ; BRANCH
014421  000000  014425                  ^O00,2$
014423  104020  003004          1$:     MOV     R2,PNUM         ; SAVE PATTERN NUMBER
014425  104021                  2$:     MOV     R2,R1           ; BUILD PATTERN NUMBER WORD IN R1
014426  110702                          SWAB    R2              ; MOVE PATTERN NUMBER TO HI BYTE
014427  101021                          BIS     R2,R1           ; SET HI BITS
014430  110202                          ROL     R2              ; ROTATE TO HI NIBBLE
014431  110202                          ROL     R2
014432  110202                          ROL     R2
014433  110202                          ROL     R2
014434  103202  007777                  BIC     #HBHINB,R2      ; CLEAR UNUSED BITS
014436  101021                          BIS     R2,R1           ; SET BITS
014437  110702                          SWAB    R2              ; MOVE TO LO BYTE
014440  101021                          BIS     R2,R1           ; SET BITS
014441  106271                          CMP     (R0)+,R1        ; SEE IF REDUNDANT PATTERN OK
014442                                  BNE     FWRD            ; IF NOT, BRANCH
014442  050000  014517                  ^O50000,FWRD
014444  104302  003004                  MOV     PNUM,R2         ; RESTORE R2
014446  103202  177760                  BIC     #LBLONB,R2      ; MAP PATTERN 16 TO PATTERN 0
014450  104622  003021                  MOV     PATPTR(R2),R2   ; POINT TO PATTERN
014452  115402                          INC     R2              ; SKIP EDC
014453  104201  000001                  MOV     #1,R1           ; R1 HAS OFFSET INTO BUFFER
                                        .DSABL  LSB
014455  104024                  XOPLP0: MOV     R2,R4           ; R4 POINTS TO LENGTH OF PATTERN
014456  104243                          MOV     (R4)+,R3        ; R3 CONTAINS LENGTH OF PATTERN
014457  106203  000001                  CMP     #1,R3           ; SEE IF PATTERN IS 1 WORD LONG
014461                                  BEQ     ONEPAX          ; IF SO, BRANCH
014461  010000  014501                  ^O10000,ONEPAX
014463  104245                  XOPLP1: MOV     (R4)+,R5        ; R5 GETS 1 WORD OF THE DATA PATTERN
014464  106275                          CMP     (R0)+,R5        ; COMPARE PATTERN WORD TO SECTOR AREA
014465                                  BNE     CMPERR          ; IF NOMATCH, BRANCH
014465  050000  014520                  ^O50000,CMPERR
014467  115201                          INC     R1              ; INCREMENT OFFSET
014470  106201  000400                  CMP     #SCTWRD+1,R1    ; SEE IF ENTIRE BUFFER COMPARED
```

```
 76 014472                                      BEQ     CSCEXT          ; IF ALL WORDS COMPARED, BRANCH
    014472  010000  014512                      ^010000,CSCEXT
 77 014474  117403                              DEC     R3              ; DECREMENT COUNT OF WORDS IN PATTERN
 78 014475                                      BNE     XOPLP1          ; IF DATA PATTERN UNFINISHED, BRANCH
    014475  050000  014463                      ^050000,XOPLP1
 79 014477                                      BR      XOPLP0          ; BRANCH
    014477  000000  014455                      ^00,XOPLP0
 80 014501  104145                      ONEPAX: MOV     (R4),R5         ; GET 1 WORD OF DATA PATTERN
 81 014502  106275                      XOPLP2: CMP     (R0)+,R5        ; COMPARE PATTERN TO SECTOR READ
 82 014503                                      BNE     CMPERR          ; IF COMPARE ERROR, BRANCH
    014503  050000  014520                      ^050000,CMPERR
 83 014505  115401                              INC     R1              ; INCREMENT NUMBER OF WORDS TO COMPARE
 84 014506  106201  000400                      CMP     #SCTWRD+1,R1    ; SEE IF ALL WORDS COMPARED
 85 014510                                      BNE     XOPLP2          ; IF INCOMPLETE, BRANCH
    014510  050000  014502                      ^050000,XOPLP2
 86 014512  114002                      CSCEXT: CLR     R2              ; NO ERRORS
 87 014513                                      POP     <R5,R4>         ; RESTORE R5,R4
    014513  104265                                                                  MOV (SP)+,R5
    014514  104264                                                                  MOV (SP)+,R4
 88 014515                                      BR      CMXEX           ; BRANCH
    014515  000000  014721                      ^00,CMXEX
 89 014517  114001                      FWRD:   CLR     R1              ; ZERO OFFSET
 90 014520                              CMPERR: POP     <R5,R4>         ; RESTORE R5,R4
    014520  104265                                                                  MOV (SP)+,R5
    014521  104264                                                                  MOV (SP)+,R4
 91 014522                                      PUSH    R0              ; SAVE POINTER TO BUFFER
    014522  100467                                                                 MOV R0,-(SP)
 92 014523  104037                              MOV     R3,R0           ; MOVE POINTER TO READ CHAIN TO R0
 93 014524                                      CALL    BLKCHK          ; SEE IF THIS IS A KNOWN BAD BLOCK
    014524  020000  002465                      ^020000,BLKCHK
 94 014526                                      POP     R0              ; RESTORE R0
    014526  104267                                                                 MOV (SP)+,R0
 95 014527                                      BCC     CMXEX           ; IF KNOWN BAD BLOCK, EXIT
    014527  040000  014721                      ^040000,CMXEX
 96 014531  104303  003015                      MOV     CHAINS,R3       ; R3 POINTS TO LINK
 97 014533                                      HARDER  12
    014533  104200  001113  001601                                              MOV     #ER12,HRQ.04
    014536  104202  107654                                                      MOV     #12!ERHARD+4000.,R2
    014540  104020  001577                                                      MOV     R2,HRQ.02
    014542  104200  014542  001576                                              MOV     #.,HRQ.01
    014545  104200  060014  001575                                              MOV     #ERRMC,HRQ.RQ
 98 014550                                      CERROR  5,#SER24        ; ECC/EDC HAD DETECTED ERROR
    014550  104200  001326  001602                                              MOV     #SER24,HRQ.05
 99 014553  115000  003003                      TST     SCR2            ; SEE IF ERROR DETECTED
100 014555                                      BNE     4$              ; IF SO, BRANCH
    014555  050000  014562                      ^050000,4$
101 014557                                      CERROR  5,#SER25        ; ECC/EDC DID NOT DETECT ERROR
    014557  104200  001354  001602                                              MOV     #SER25,HRQ.05
102 014562                              4$:     ERRORC  <S.LETR(R4),RW.LOW(R3),RW.HI(R3)>
    014562  104640  000005  001603                                              MOV     S.LETR(R4),HRQ.06
    014565  104630  000003  001604                                              MOV     RW.LOW(R3),HRQ.07
    014570  104630  000004  001605                                              MOV     RW.HI(R3),HRQ.08
103 014573                                      ERRORC  <U.PARM(R5),#RBNTXT>
    014573  104650  000046  001606                                              MOV     U.PARM(R5),HRQ.09
    014576  104200  005472  001607                                              MOV     #RBNTXT,HRQ.10
104 014601                                      ERRORC  <U.RBN(R5),U.RBN+1(R5),RW.ANG(R3)>
    014601  104650  000055  001610                                              MOV     U.RBN(R5),HRQ.11
```

```
                                                                        MOV     U.RBN+1(R5),HRQ.12
                                                                        MOV     RW.ANG(R3),HRQ.13
105 014612                              ERRORC  <RW.CMD(R3),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
    014612  104630  000005  001613                                      MOV     RW.CMD(R3),HRQ.14
    014615  104650  000066  001614                                      MOV     U.CGRP(R5),HRQ.15
    014620  104650  000064  001615                                      MOV     U.CCYL(R5),HRQ.16
    014623  104650  000065  001616                                      MOV     U.CCYL+1(R5),HRQ.17
106 014626  106201  000366              CMP     #246.,R1        ; SEE IF IN LAST 9 WORDS
107 014630                              BPL     1$              ; IF NOT, BRANCH
    014630  030000  014643              ^030000,1$
108 014632  104013                      MOV     R1,R3           ; TO COMPUTE OFFSET OF ERROR
109 014633  107203  000367              SUB     #247.,R3        ; R3 HAS UNADJUSTED ERROR POSITION
110 014635  105203  000003              ADD     #3,R3           ; R3 HAS ADJUSTED ERROR POSITION
111 014637  107037                      SUB     R3,R0           ; ADJUST POINTER
112 014640  117407                      DEC     R0              ; COMPARE AUTO-INCREMENTED, SO ADJUST
113 014641                              BR      3$              ; PRINT ERROR
    014641  000000  014660              ^00,3$
114 014643  106201  000004      1$:     CMP     #4,R1           ; SEE IF IN FIRST 4 WORDS
115 014645                              BCC     2$              ; IF SO, BRANCH
    014645  040000  014655              ^040000,2$
116 014647  107207  000004              SUB     #4,R0           ; LOOK BACK FOUR WORDS
117 014651  104203  000003              MOV     #3,R3           ; OFFSET OF ERROR IN PRINTOUT IS 3
118 014653                              BR      3$              ; BRANCH
    014653  000000  014660              ^00,3$
119 014655  107017              2$:     SUB     R1,R0           ; BACK UP POINTER
120 014656  104013                      MOV     R1,R3           ; R3 IS OFFSET OF ERROR IN PRINTOUT
121 014657  117407                      DEC     R0              ; TO POINT TO START OF BUFFER
122 014660                      3$:     ERRORC  <PNUM,R1,R3,(R0)+,(R0)+,(R0)+,(R0)+,(R0)+,(R0)+,(R0)+,(R0)+>
    014660  104300  003004  001617                                      MOV     PNUM,HRQ.18
    014663  104010  001620                                              MOV     R1,HRQ.19
    014665  104030  001621                                              MOV     R3,HRQ.20
    014667  104270  001622                                              MOV     (R0)+,HRQ.21
    014671  104270  001623                                              MOV     (R0)+,HRQ.22
    014673  104270  001624                                              MOV     (R0)+,HRQ.23
    014675  104270  001625                                              MOV     (R0)+,HRQ.24
    014677  104270  001626                                              MOV     (R0)+,HRQ.25
    014701  104270  001627                                              MOV     (R0)+,HRQ.26
    014703  104270  001630                                              MOV     (R0)+,HRQ.27
    014705  104270  001631                                              MOV     (R0)+,HRQ.28
123 014707                              ERRORC  <(R0)+,(R0)+,(R0)+,(R0)+>
    014707  104270  001632                                              MOV     (R0)+,HRQ.29
    014711  104270  001633                                              MOV     (R0)+,HRQ.30
    014713  104270  001634                                              MOV     (R0)+,HRQ.31
    014715  104270  001635                                              MOV     (R0)+,HRQ.32
124 014717                              ENDERR  0
    014717  114000  003013                                              CLR     ERRPOS          ; CLEAR THE POSITION
125 014721  104207  014726      CMXEX:  MOV     #LSTMOD,R0      ; LAST MODULE WILL BE CALLED
126 014723  114001                      CLR     R1              ; CALL IMMIDATELY
127 014724                              BR      JMPRET          ; RETURN TO SEQUNCR
    014724  000000  003746              ^00,JMPRET
```

```
                                .SBTTL  ********** OVERLAY MODULE LSTMOD - CLEANUP MODULE BEFORE SETUP
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                LSTMOD:
                                ;
                                ;       LOOP CONTROL FOR ECC, EDC AND DATA COMPARE MODULES.  ALSO LAST
                                ;       MODULE BEFORE SETUP (USUALLY)
                                ;
                                        .ENABL  LSB
014726  114002                          CLR     R2                      ; ASSUME NO ERRORS
014727  104657  000021                  MOV     U.NSEC(R5),R0           ; GET NUMBER OF SECTORS HANDLED SO FAR
014731  115407                          INC     R0                      ; INCREMENT COUNT
014732  100657  000021                  MOV     R0,U.NSEC(R5)           ; SAVE
014734  104701  166057                  MOV     @CHAINS,R1              ; SEE IF LAST BUFFER LAST READ
014736                                  ASSUME  RW.CPT,0                ; ASSUME FULL FLAG, BUFFER POINTER OFFSET ZERO
014736                                  BMI     1$                      ; IF SO, BRANCH
014736  070000  014751                  ^O070000,1$
014740                                  ASSUME  EOC,100000              ; ASSUME END-OF-CHAIN SIGN BIT
014740  103201  140000                  BIC     #UNADDR,R1              ; CLEAR UNUSED ADDRESSING BITS
014742  104010  003015                  MOV     R1,CHAINS               ; CHAINS NOW POINTS TO NEXT LINK
014744  104207  012134                  MOV     #SECCHK,R0              ; SECTOR CHECK NEXT MODULE
014746  114001                          CLR     R1                      ; IMMIDATE CALL
014747                                  BR      4$                      ; EXIT
014747  000000  015005                  ^O00,4$
014751  104653  000046          1$:     MOV     U.PARM(R5),R3           ; GET UNIT PARAMETERS
014753  103203  000200                  BIC     #RBNBN,R3               ; IF HANDLING AN RBN, CLEAR IT
014755  100653  000046                  MOV     R3,U.PARM(R5)           ; SAVE
014757  105647  000010                  ADD     S.MEGR(R4),R0           ; GET MEGABYTE COUNT
014761  106207  003642                  CMP     #1954.,R0               ; SEE IF ONE MEGABYTE TRANSFERED
014763                                  BPL     3$                      ; IF NOT, BRANCH
014763  030000  015000                  ^O030000,3$
014765  107207  003642                  SUB     #1954.,R0               ; ZERO COUNT
014767  104200  000001  001577          MOV     #1,HRQ.02               ; REPORT 1 MEGABYTE READ
014772  114000  001600                  CLR     HRQ.03                  ; NO WRITE MEGABYTES REPORTED
014774  104202  060011                  MOV     #T4MXFR,R2              ; SET UP FOR MEGABIT REPORT
014776  104020  001575                  MOV     R2,HRQ.RQ               ; FLAG AS NON-ERROR
015000  100647  000010          3$:     MOV     R0,S.MEGR(R4)           ; SAVE COUNT
015002  104207  005134                  MOV     #SETUP,R0               ; SETUP IS NEXT MODULE CALLED
015004  104051                          MOV     R5,R1                   ; DEFERRED CALL TO NEXT MODULE
015005                          4$:     BR      JMPRET                  ; RETURN TO SEQUENCER
015005  000000  003746                  ^O00,JMPRET
                                        .DSABL  LSB
```

```
                                .SBTTL  ********** OVERLAY MODULE REVCT - REVECTORED SECTOR HANDLING
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                ;
015007                          REVCT:
                                ;       REVECTOR A SECTOR WITH A HEADER NOT FOUND OR A SECONDARY REVECTOR
                                        .ENABL  LSB
015007  104657  000046                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
015011  102207  000400                  BIT     #REVEC,R0       ; SEE IF REVECTOR ALLREADY IN PROGRESS
015013                                  BNE     1$              ; IF SO, BRANCH
015013  050000  015074                  ^O50000,1$
                                .SBTTL  REVSUP - SETUP THE REVECTOR OPERATION (TO READ RCT)
                                ;REVSUP
                                ;
                                ;       INITILIZE ALL PARAMETERS FOR READING THE RCT TO FING THE RBN THAT
                                ;       A HEADER HAS BEEN REVECTORED TO
                                ;
015015  101207  000400                  BIS     #REVEC,R0       ; FLAG AS REVECTOR IN PROGRESS
015017  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE PARAMETERS
015021  104207  005523                  MOV     #RCTL$,R0       ; POINT OT RCT LBN STRING
015023  100647  000005                  MOV     R0,S.LETR(R4)   ; SAVE
015025  104657  000061                  MOV     U.RWER(R5),R0   ; GET READ/WRITE ERROR TYPE
015027  100657  000062                  MOV     R0,U.RVER(R5)   ; SAVE FOR REVECTOR INFORMATION
015031  104641  000007                  MOV     S.SCHR(R4),R1   ; R1 POINTS TO SUBUNIT CHARACTERISTICS
015033  114007                          CLR     R0              ; USE R0 TO INITILIZE VALUES
015034  100657  000055                  MOV     R0,U.RBN(R5)    ; START WITH RBN ZERO
015036  100657  000056                  MOV     R0,U.RBN+1(R5)  ; START WITH RBN ZERO
015040  100657  000060                  MOV     R0,U.CCOP(R5)   ; ON ORIGINAL COPY OF RCT
015042  115407                          INC     R0              ; R0 IS NOW 1
015043  100657  000022                  MOV     R0,U.MSEC(R5)   ; ONLY READ 1 SECTOR A^ A TIME
015045  115407                          INC     R0              ; R0 IS NOW 2
015046  105617  000012                  ADD     LBNHST(R1),R0   ; R0 POINTS TO 1ST REVECTOR INFORMATION SECTOR
015050  100657  000053                  MOV     R0,U.CBN(R5)    ; SAVE
015052  104617  000013                  MOV     LBNHST+1(R1),R0 ; R0 HAS HI FIRST REV INFO SECTOR
015054                                  BCC     5$              ; IF NO CARY, BRANCH
015054  040000  015057                  ^O40000,5$
015056  115407                          INC     R0              ; PROPOGATE CARRY
015057  103207  170000          5$:     BIC     #^CHBHINB,R0    ; CLEAR SUBUNIT BITS
015061  100657  000054                  MOV     R0,U.CBN+1(R5)  ; SAVE
015063  114002                          CLR     R2              ; NO ERRORS
015064  104201  177777                  MOV     #-1,R1          ; START READ ATTEMPT RETRIES AT 0
015066  100651  000012                  MOV     R1,U.RWTO(R5)   ; SAVE
015070  104207  015642                  MOV     #SEEK,R0        ; SEEK IS NEXT MODULE
015072                                  BR      4$              ; EXIT
015072  000000  015133                  ^O0,4$
015074  104651  000024          1$:     MOV     U.CSEC(R5),R1   ; GET NUMBER OF SECTORS R/W SO FAR
015076  105651  000051                  ADD     U.MBN(R5),R1    ; ADD STARTING SECTOR OF OPERATION
015100  104010  002760                  MOV     R1,CURBN        ; MOVE TO TEMP STORAGE
015102  104651  000052                  MOV     U.MBN+1(R5),R1  ; GET HI STARTING SECTOR
015104                                  BCC     2$              ; IF NO CARRY, BRANCH
015104  040000  015107                  ^O40000,2$
015106  115401                          INC     R1              ; PROPOGATE CARRY
015107  104010  002761          2$:     MOV     R1,CURBN+1      ; SAVE
```

```
     79 015111  104301  003003                    MOV     SCR2,R1         ; SEE IF SECTOR READ IS OK
     80 015113                                     BEQ     6$              ; IF SO, BRANCH
        015113  010000  015121                     ^O10000,6$
     81 015115                                     CALL    REVSOK          ; FIND NEXT COPY TO READ
        015115  020000  015135                     ^O20000,REVSOK
     82 015117                                     BR      4$              ; EXIT
        015117  000000  015133                     ^O00,4$
     83 015121                             6$:     CALL    SEARCH          ; SEARCH THE SECTOR TO FIND THE LBN
        015121  020000  015255                     ^O20000,SEARCH
     84 015123  115001                             TST     R1              ; SEE IF LBN FOUND
     85 015124                                     BEQ     4$              ; LBN FOUND, BRANCH AND READ RBN
        015124  010000  015133                     ^O10000,4$
     86 015126  115002                             TST     R2              ; SEE IF NULL FLAG FOUND (REVECTOR NOT FOUND)
     87 015127                                     BNE     4$              ; IF NOT, BRANCH
        015127  050000  015133                     ^O50000,4$
     88 015131                                     CALL    NXTRCT          ; READ NEXT RCT SECTOR
        015131  020000  015432                     ^O20000,NXTRCT
     89 015133                             4$:     BR      JMPRET          ; RETURN CONTROL TO SEQUENCER
        015133  000000  003746                     ^O00,JMPRET
     90                                            .DSABL  LSB
```

REVSOK - SEE IF THE REVECTOR INFO SECTOR JUST READ IS OK

```
                                .SBTTL  REVSOK - SEE IF THE REVECTOR INFO SECTOR JUST READ IS OK
015135                          REVSOK:
                                ;
                                ;       GO TO THE NEXT RCT COPY
                                ;
015135  104651  000060          MOV     U.CCOP(R5),R1   ; GET NUMBER OF COPIES TRIED SO FAR
015137  115401                  INC     R1              ; TRY ANOTHER COPY
015140  106651  000057          CMP     U.COPY(R5),R1   ; CHECK AGAINST MAX
015142                          BPL     5$              ; IF ALL COPIES UNTRIED, BRANCH
015142  030000  015221          ^030000,5$
015144                          HARDER  40
015144  104200  002705  001601                                  MOV     #ER40,HRQ.04
015147  104202  107710                                          MOV     #40!ERHARD+4000.,R2
015151  104020  001577                                          MOV     R2,HRQ.02
015153  104200  015153  001576                                  MOV     #.,HRQ.01
015156  104200  060014  001575                                  MOV     #ERRMC,HRQ.RQ
015161  104651  000062          MOV     U.RVER(R5),R1   ; GET ORIGINAL ERROR TYPE
015163  106201  000003          CMP     #3,R1           ; SEE IF REVECTORED BLOCK
015165                          BNE     2$              ; IF NOT, BRANCH
015165  050000  015174          ^050000,2$
015167                          CERROR  5,#SER26        ; FLAG ERROR
015167  104200  003035  001602                                  MOV     #SER26,HRQ.05
015172                          BR      36$             ; EXIT
015172  000000  015177          ^00,36$
015174                  2$:     CERROR  5,#SER32        ; FLAG ERROR HEADER COMPARE ERROR
015174  104200  003051  001602                                  MOV     #SER32,HRQ.05
015177                  36$:    ERRORC  <U.CBN(R5),U.CBN+1(R5),CURBN,CURBN+1>
015177  104650  000053  001603                                  MOV     U.CBN(R5),HRQ.06
015202  104650  000054  001604                                  MOV     U.CBN+1(R5),HRQ.07
015205  104300  002760  001605                                  MOV     CURBN,HRQ.08
015210  104300  002761  001606                                  MOV     CURBN+1,HRQ.09
015213                          ENDERR  0
015213  114000  003013                                          CLR     ERRPOS          ; CLEAR THE POSITION
015215                          CALL    RVFAIL          ; CLEAR ALL BITS AND DO A FAIL EXIT
015215  020000  015603          ^020000,RVFAIL
015217                          BR      7$              ; EXIT
015217  000000  015253          ^00,7$
015221  100651  000060  5$:     MOV     R1,U.CCOP(R5)   ; SAVE COPY COUNT
015223  104647  000007          MOV     S.SCHR(R4),R0   ; R0 POINTS TO SUBUNIT PARAMETERS
015225  104677  000014          MOV     RCTCSZ(R0),R0   ; R0 IS RCT COPY SIZE
015227  105657  000053          ADD     U.CBN(R5),R0    ; ADD CURRENT SECTOR
015231  100657  000053          MOV     R0,U.CBN(R5)    ; SAVE
015233                          BCC     6$              ; IF NO CARRY, BRANCH
015233  040000  015242          ^040000,6$
015235  104657  000054          MOV     U.CBN+1(R5),R0  ; GET HI CURRENT BLOCK NUMBER
015237  115407                  INC     R0              ; INCREMENT COUNT
015240  100657  000054          MOV     R0,U.CBN+1(R5)  ; SAVE
015242                  6$:     CALL    ENABLE
015242  020000  005036          ^020000,ENABLE
015244  104203  177777          MOV     #-1,R3          ; START READ ATTEMPT RETRIES AT 0
015246  100653  000012          MOV     R3,U.RWTO(R5)   ; SAVE
015250  104207  015642          MOV     #SEEK,R0        ; SEEK IS NEXT MODULE
015252  114002                  CLR     R2              ; NO ERRORS
015253                  7$:     RETURN                  ; RETURN TO CALLING PROGRAM
015253  000000  000000          ^00,0
```

```
                                .SBTTL  SEARCH - TRY TO FIND THE LBN IN THE RCT SECTOR JUST READ
 015255                         SEARCH:
                                ;
                                ;       SEARCH THE RCT SECTOR JUST READ TO FIND THE LBN OR THE NULL ENTRY
                                ;
 015255  104307  003015                 MOV     CHAINS,R0       ; R0 POINTS TO LINK (NODE) IN READ CHAIN
 015257  104677  000002                 MOV     RW.BUF(R0),R0   ; R0 NOW POINTS TO BUFFER
 015261  114001                         CLR     R1              ; R1 IS RBN NUMBER OFFSET WITHIN THE SECTOR
 015262  104672  000001         1$:     MOV     1(R0),R2        ; GET RCT CODE
 015264                                 BMI     6$              ; IF NULL POINTER, ENTIRE TABLE EXHAUSTED, BRANCH
 015264  070000  015362                 ^O70000,6$
 015266  102202  020000                 BIT     #020000,R2      ; SEE IF IT IS A USED RBN
 015270                                 BEQ     5$              ; IF NOT, BRANCH
 015270  010000  015335                 ^O10000,5$
 015272  103202  170000                 BIC     #^CHBHINB,R2    ; CLEAR CODE
 015274  106302  002761                 CMP     CURBN+1,R2      ; SEE IF HI ORDER MATCHES
 015276                                 BNE     5$              ; IF NOT, BRANCH
 015276  050000  015335                 ^O50000,5$
 015300  106170  002760                 CMP     (R0),CURBN      ; SEE IF LO ORDER MATCHES
 015302                                 BNE     5$              ; IF NOT, BRANCH
 015302  050000  015335                 ^O50000,5$
 015304  105651  000055                 ADD     U.RBN(R5),R1    ; ADD RUNNING RBN TO OFFSET
 015306  100651  000055                 MOV     R1,U.RBN(R5)    ; SAVE
 015310                                 BCC     2$              ; IF NO CARRY, BRANCH
 015310  040000  015317                 ^O40000,2$
 015312  104651  000056                 MOV     U.RBN+1(R5),R1  ; GET HI ORDER RBN
 015314  115401                         INC     R1              ; PROPOGATE CARRY
 015315  100651  000056                 MOV     R1,U.RBN+1(R5)  ; SAVE
 015317  104657  000046         2$:     MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
 015321  101207  000200                 BIS     #RBNBN,R0       ; FLAG ALL ROUTINES THAT THIS IS AN RBN
 015323  100657  000046                 MOV     R0,U.PARM(R5)   ; SAVE
 015325                                 CALL    RVFAIL          ; FAIL EXIT DOES WHAT WE WANT, JUST CLEAR R1 AND R2
 015325  020000  015603                 ^O20000,RVFAIL
 015327  114001                         CLR     R1              ; FOUND IT
 015330  114002                         CLR     R2              ; NO ERRORS
 015331  100652  000021                 MOV     R2,U.NSEC(R5)   ; SAVE
 015333                                 BR      7$              ; EXIT
 015333  000000  015430                 ^O00,7$
 015335  105207  000002         5$:     ADD     #2,R0           ; POINT TO NEXT RBN RECORD
 015337  115401                         INC     R1              ; INCREMENT RBN OFFSET
 015340  106201  000200                 CMP     #128.,R1        ; SEE IF ALL RECORDS TRIED
 015342                                 BNE     1$              ; IF NOT, BRANCH
 015342  050000  015262                 ^O50000,1$
 015344  114002                         CLR     R2              ; TO SIGNAL CALLING ROUTINE TO READ NEXT SECTOR
 015345  105651  000055                 ADD     U.RBN(R5),R1    ; ADD OLD RBN TO 128
 015347  100651  000055                 MOV     R1,U.RBN(R5)    ; SAVE
 015351                                 BCC     7$              ; IF NO CARRY, EXIT
 015351  040000  015430                 ^O40000,7$
 015353  104651  000056                 MOV     U.RBN+1(R5),R1  ; GET HI ORDER RBN
 015355  115401                         INC     R1              ; PROPOGATE CARRY
 015356  100651  000056                 MOV     R1,U.RBN+1(R5)  ; SAVE
 015360                                 BR      7$              ; EXIT
 015360  000000  015430                 ^O00,7$
 015362                         6$:     HARDER  41
 015362  104200  002775  001601                                         MOV     #ER41,HRQ.04
 015365  104202  107711                                                 MOV     #41!ERHARD+4000.,R2
 015367  104020  001577                                                 MOV     R2,HRQ.02
```

```
    015371  104200  015371  001576                                          MOV     #.,HRQ.01
    015374  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
 45 015377  104651  000062                  MOV     U.RVER(R5),R1   ; GET ORIGINAL ERROR
 46 015401  106201  000003                  CMP     #3,R1           ; SEE IF REVECTORED BLOCK
 47 015403                                  BNE     32$             ; IF NOT, BRANCH
    015403  050000  015412                  ^O50000,32$
 48 015405                                  CERROR  5,#SER26        ; REVECTORED SECTOR
    015405  104200  003035  001602                                          MOV     #SER26,HRQ.05
 49 015410                                  BR      12$             ; EXIT
    015410  000000  015415                  ^O00,12$
 50 015412                          32$:    CERROR  5,#SER32        ; REPORT HEADER COMPARE ERROR
    015412  104200  003051  001602                                          MOV     #SER32,HRQ.05
 51 015415                          12$:    ERRORC  <CURBN,CURBN+1>
    015415  104300  002760  001603                                          MOV     CURBN,HRQ.06
    015420  104300  002761  001604                                          MOV     CURBN+1,HRQ.07
 52 015423                                  ENDERR  0
    015423  114000  003013                                                  CLR     ERRPOS          ; CLEAR THE POSITION
 53 015425                           9$:    CALL    RVFAIL          ; CLEAR ALL BITS, FAIL EXIT
    015425  020000  015603                  ^O20000,RVFAIL
 54 015427  104051                          MOV     R5,R1           ; FLAG AS ERROR
 55 015430                           7$:    RETURN                  ; RETURN TO CALLING PROGRAM
    015430  000000  000000                  ^O00,0
```

```
                                        .SBTTL  NXTRCT - GET THE NEXT RCT SECTOR TO SEARCH
015432                                  NXTRCT:
                                        ;
                                        ;       INCREMENT CBN BY 1 AFTER SUBTACTING THE NUNMBER OF COPIES*RCT SIZE
                                        ;       THAT WAS SEARCHED THROUGH ON THE LAST PASS
                                        ;
015432  104652  000053                        MOV     U.CBN(R5),R2    ; R2 HAS LO ORDER RCT BN
015434  104653  000054                        MOV     U.CBN+1(R5),R3  ; R3 HAS HI ORDER RCT BN
015436  104657  000060                        MOV     U.CCOP(R5),R0   ; GET NUMBER OF COPIES THAT HAVE BEEN READ
015440                                      BEQ     3$              ; IF NO COPIES, BRANCH
015440  010000  015454                      ^010000,3$
015442  104641  000007                      MOV     S.SCHR(R4),R1   ; R1 POINTS TO SUBUNIT CHARACTERISTICS
015444  107612  000014              1$:     SUB     RCTCSZ(R1),R2   ; SUBTRACT FROM LO ORDER BN
015446                                      BCC     2$              ; IF NO BORROW, BRANCH
015446  040000  015451                      ^040000,2$
015450  117403                              DEC     R3              ; PROPOGATE BORROW
015451  117407                      2$:     DEC     R0              ; DECREMENT COUNT
015452                                      BNE     1$              ; IF NO CARRY, BRANCH
015452  050000  015444                      ^050000,1$
015454  105202  000001              3$:     ADD     #1,R2           ; INCREMENT CBN BY 1
015456                                      BCC     4$              ; IF NO CARRY, BRANCH
015456  040000  015461                      ^040000,4$
015460  115403                              INC     R3              ; PROPOGATE CARRY
015461  100652  000053              4$:     MOV     R2,U.CBN(R5)    ; SAVE LO ORDER
015463  100653  000054                      MOV     R3,U.CBN+1(R5)  ; SAVE HI ORDER
015465  104020  002760                      MOV     R2,CURBN        ; MOVE TO CALC AREA
015467  104030  002761                      MOV     R3,CURBN+1      ; MOVE TO CALC AREA
                                        ;                             NOTE R0 MUST BE ZERO FOR CALC SETUP
015471                                      CALL    CALC            ; FIND CYL THAT NEXT SEC IS ON
015471  020000  002604                      ^020000,CALC
015473  104647  000007                      MOV     S.SCHR(R4),R0   ; POINT TO SUBUNIT CHARACTERISTICS
015475  106670  000001  002765              CMP     HICYL(R0),CYL+1 ; SEE IF IN XBN AREA
015500                                      BCS     5$              ; IF SO, BRANCH
015500  040000  015504                      ^040000,.+3
015502  000000  015515                      ^00,5$
015504                                      BNE     6$              ; IF NOT, BRANCH
015504  050000  015570                      ^050000,6$
015506  106670  000000  002764              CMP     LBNCYL(R0),CYL  ; SEE IF IN XBN AREA
015511                                      BEQ     5$              ; IF SO, BRANCH
015511  010000  015515                      ^010000,5$
015513                                      BCC     6$              ; IF NOT, BRANCH
015513  040000  015570                      ^040000,6$
015515                              5$:     CALL    RVFAIL          ; REVECTOR FAILED
015515  020000  015603                      ^020000,RVFAIL
015517                                      HARDER  4               ; REPORT CORRUPTED RCT
015517  104200  000271  001601                                              MOV     #ER4,HRQ.04
015522  104202  107644                                                      MOV     #4!ERHARD+4000.,R2
015524  104020  001577                                                      MOV     R2,HRQ.02
015526  104200  015526  001576                                              MOV     #.,HRQ.01
015531  104200  060014  001575                                              MOV     #ERRMC,HRQ.RQ
015534  104651  000062                      MOV     U.RVER(R5),R1   ; GET ORIGINAL ERROR
015536  106201  000003                      CMP     #3,R1           ; SEE IF REVECTOR
015540                                      BNE     7$              ; IF NOT, BRANCH
015540  050000  015547                      ^050000,7$
015542                                      CERROR  5,#SER26        ; REPORT REVECTOR
015542  104200  003035  001602                                              MOV     #SER26,HRQ.05
015545                                      BR      8$              ; BRANCH
```

```
    015545  000000  015552                  ^O0,8$
 40 015547                          7$:     CERROR  5,#SER32        ; REPORT HEADER COMPARE ERROR
    015547  104200  003051  001602                                          MOV     #SER32,HRQ.05
 41 015552                          8$:     ERRORC  <CURBN,CURBN+1,U.CBN(R5),U.CBN+1(R5)>
    015552  104300  002760  001603                                          MOV     CURBN,HRQ.06
    015555  104300  002761  001604                                          MOV     CURBN+1,HRQ.07
    015560  104650  000053  001605                                          MOV     U.CBN(R5),HRQ.08
    015563  104650  000054  001606                                          MOV     U.CBN+1(R5),HRQ.09
 42 015566                                  BR      9$              ; REPORT ERROR
    015566  000000  015601                  ^O0,9$
 43 015570  114002                  6$:     CLR     R2              ; NO ERRORS
 44 015571                                  CALL    ENABLE          ; ENABLE ERROR RECOVERY
    015571  020000  005036                  ^O20000,ENABLE
 45 015573  104203  177777                  MOV     #-1,R3          ; START READ ATTEMPT RETRIES AT 0
 46 015575  100653  000012                  MOV     R3,U.RWTO(R5)   ; SAVE
 47 015577  104207  015642                  MOV     #SEEK,R0        ; NEXT MODULE IS SEEK
 48 015601                          9$:     RETURN
    015601  000000  000000                  ^O0,0
```

```
      1                                  .SBTTL  RVFAIL - CLEAR ALL REVECTOR BITS, AND CALL SETUP NEXT
      2 015603                           RVFAIL:
      3                                  ;
      4                                  ;       CLEAR ALL REVECTOR BITS, AND RESTORE U.CBN TO WHAT IT WAS
      5                                  ;
      6 015603  104651  000046                   MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
      7 015605  103201  000400                   BIC     #REVEC,R1       ; CLEAR ALL REVECTOR BITS
      8 015607  100651  000046                   MOV     R1,U.PARM(R5)   ; SAVE
      9 015611  104201  000001                   MOV     #1,R1           ; FLAG THAT A SECTOR HAS BEEN READ (SKIPPED)
     10 015613  100651  000021                   MOV     R1,U.NSEC(R5)   ; SAVE
     11 015615  104651  000024                   MOV     U.CSEC(R5),R1   ; GET NUMBER OF SECTORS
     12 015617  105651  000051                   ADD     U.MBN(R5),R1    ; ADD STARTING SECTOR NUMBER
     13 015621  100651  000053                   MOV     R1,U.CBN(R5)    ; RESTORE CBN
     14 015623  104651  000052                   MOV     U.MBN+1(R5),R1  ; GET HI MBN
     15 015625                                   BCC     1$              ; IF NO CARRY, BRANCH
        015625  040000  015630                   ^O040000,1$
     16 015627  115401                          INC     R1              ; PROPOGATE CARRY
     17 015630  100651  000054          1$:     MOV     R1,U.CBN+1(R5)  ; SAVE HI IN CBN
     18 015632  104207  005517                   MOV     #L$,R0          ; GET LBN STRING
     19 015634  100647  000005                   MOV     R0,S.LETR(R4)   ; SAVE
     20 015636  104207  005134                   MOV     #SETUP,R0       ; SETUP NEXT ROUTINE
     21 015640                                   RETURN                  ; RETURN TO CALLING PROGRAM
        015640  000000  000000                   ^O0,0
```

```
                                .SBTTL  ********** OVERLAY MODULE SEEK - IF NECESSARY, ISSUE SEEK
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                ;
015642                          SEEK:
                                ;
                                ;       SEEK ISSUES A SEEK TO THE CYLINDER SPECIFIED IN U.CCYL(R5) (2 WORDS)
                                ;       AND GROUP GIVEN IN U.CGRP(R5) (1 WORD).
                                ;       IT THEN RETURNS WITH A DEFERED CALL TO SEKTST (TEST SEEK) TO SEE IF
                                ;       THE SEEK COMPLETED SUCCESSFULLY.  IF SO, THE READ WRITE MODULE IS
                                ;       THEN CALLED
                                ;
                                        .ENABL  LSB
015642  104653  000046                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
015644  102203  002000                  BIT     #SEKINP,R3      ; SEE IF SEEK IS ALLREADY IN PROGRESS
015646                                  BNE     3$              ; IF SO, BRANCH
015646  050000  015654                  ^O50000,3$
015650  104201  177777                  MOV     #-1,R1          ; FOR ZEROING FOLLOWING WORDS
015652  100651  000007                  MOV     R1,U.SRTY(R5)   ; SAVE
015654  104653  000047          3$:     MOV     U.RCOV(R5),R3   ; GET RECOVERY WORD
015656  102203  002000                  BIT     #RCBREQ,R3      ; SEE IF RECALIBRATION REQUESTED
015660                                  BEQ     4$              ; IF NOT, BRANCH
015660  010000  015667                  ^O10000,4$
015662  104207  016635                  MOV     #RECALB,R0      ; RECALIBRATE NEXT MODULE
015664  114002                          CLR     R2              ; NO ERRORS
015665                                  BR      SEKOUT          ; EXIT
015665  000000  016052                  ^O00,SEKOUT
015667  104652  000007          4$:     MOV     U.SRTY(R5),R2   ; GET SEEK RETRIES
015671  115402                          INC     R2              ; ADJUST FOR TEST
015672                                  BEQ     5$              ; IF FIRST TIME, BRANCH
015672  010000  016013                  ^O10000,5$
015674  106202  000002                  CMP     #2,R2           ; SEE IF TRIED MAX NUM OF TIMES
015676                                  BCC     1$              ; IF NOT, BRANCH
015676  040000  015770                  ^O40000,1$
015700                                  DEVFTL  14,<S.LETR(R4),U.CBN(R5),U.CBN+1(R5)>
015700  104200  001435  001601                                          MOV     #ER14,HRQ.04
015703  104640  000005  001602                                          MOV     S.LETR(R4),HRQ.05
015706  104650  000053  001603                                          MOV     U.CBN(R5),HRQ.06
015711  104650  000054  001604                                          MOV     U.CBN+1(R5),HRQ.07
015714  104202  047656                                                  MOV     #14!FTLDEV+4000.,R2
015716  104020  001577                                                  MOV     R2,HRQ.02
015720  104200  015720  001576                                          MOV     #.,HRQ.01
015723  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
015726                                  ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5),U.RBN+1(R5),GROUP,CYL,CYL+1>
015726  104650  000046  001605                                          MOV     U.PARM(R5),HRQ.08
015731  104200  005472  001606                                          MOV     #RBNTXT,HRQ.09
015734  104650  000055  001607                                          MOV     U.RBN(R5),HRQ.10
015737  104650  000056  001610                                          MOV     U.RBN+1(R5),HRQ.11
015742  104300  002766  001611                                          MOV     GROUP,HRQ.12
015745  104300  002764  001612                                          MOV     CYL,HRQ.13
015750  104300  002765  001613                                          MOV     CYL+1,HRQ.14
015753                                  ENDERR  0
015753  114000  003013                                                  CLR     ERRPOS          ; CLEAR THE POSITION
```

```
                                        MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
                                        BIC     #SEKINP,R3      ; CLEAR SEEK IN PROGRESS BIT
                                        MOV     R3,U.PARM(R5)   ; SAVE
                                        CALL    GORCLB          ; RECALIBRATE DRIVE
                                        ^O20000,GORCLB
                                        MOV     R5,R1           ; DEFFERED CALL
                                        BR      SEKEXT          ; EXIT
                                        ^O0,SEKEXT
                                1$:     REPSFT  .,SEEK          ; REPORT SEEK ERROR
                                                                        CLR     HRQ.02
                                                                        CLR     HRQ.03
                                                                        MOV     #1,HRQ.04
                                                                                MOV R0,-(SP)
                                                                        MOV     U.UNUM(R5),R0
                                                                        ADD     U.SUBU(R5),R0
                                                                        MOV     R0,HRQ.01
                                                                        MOV     #T4SOFT,R0
                                        ^O20000,HOSTRQ
                                                                                MOV (SP)+,R0
                                5$:     CALL    TSTNEC          ; SEE IF SEEK IS NECESSARY
                                        ^O20000,TSTNEC
                                        TST     R3              ; SEE IF SEEK IS NECESSARY
                                        BEQ     NOSEEK          ; IF ZERO (NO SEEK NECESSARY), BRANCH
                                        ^O10000,NOSEEK
                                        CALL    ISUSEK          ; ISSUE SEEK
                                        ^O20000,ISUSEK
                                        TST     R2              ; SEE IF ERROR OCCURRED
                                        BNE     SEKEXT          ; IF ERROR, BRANCH
                                        ^O50000,SEKEXT
                                        MOV     U.SRTY(R5),R1   ; GET SEEK TIMEOUT
                                        INC     R1              ; INCREMENT COUNT
                                        MOV     R1,U.SRTY(R5)   ; SAVE
                                        MOV     U.MSTO(R5),R0   ; MOVE TIMEOUT TO PARAMETERS
                                        MOV     R0,U.TIMH(R5)   ; INITILIZE TIMEOUT VALUE
                                ;       NOTE:   R2 IS ZERO HERE
                                        MOV     R2,U.TIML(R5)   ; SAVE
                                        MOV     R5,R1           ; DEFERRED CALL TO SEKTST
                                        MOV     #SEKTST,R0      ; SEKTST NEXT TO CALL
                                        CALL    DSABLE          ; DISABLE ERROR RECOVERY DURING SEEK
                                        ^O20000,DSABLE
                                        BR      SEKEXT          ; BRANCH
                                        ^O0,SEKEXT
                                NOSEEK: MOV     #BUILDP,R0      ; READ/WRITE ROUTINE NEXT
                                        CLR     R2              ; NO ERRORS
                                SEKOUT: CLR     R1              ; IMMIDATE CALL
                                SEKEXT: BR      JMPRET          ; RETURN TO CALLING PROGRAM
                                        ^O0,JMPRET
                                        .DSABL  LSB
```

```
                                        .SBTTL  TSTNEC - TEST TO SEE IF SEEK IS NECESSARY
016055                                  TSTNEC:
                                        ;
                                        ;       TSTNEC TESTS TO SEE IF A SEEK IS NECESSARY.  IF NOT, R3 IS RETURNED
                                        ;       AS ZERO, NONZERO OTHERWISE.
                                        ;
016055  104657  000046                          MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
016057  102207  000200                          BIT     #RBNBN,R0       ; SEE IF BLOCK REVECTORED
016061                                          BEQ     2$              ; IF NOT, BRANCH
016061  010000  016073                          ^010000,2$
016063  104650  000055  002760                  MOV     U.RBN(R5),CURBN ; MOVE RBN TO CALCULATION AREA
016066  104650  000056  002761                  MOV     U.RBN+1(R5),CURBN+1     ; MOVE RBN TO CALCULATION AREA
016071                                          BR      3$              ; BRANCH
016071  000000  016101                          ^00,3$
016073  104650  000053  002760  2$:     MOV     U.CBN(R5),CURBN ; MOVE LBN TO CALCULATION AREA
016076  104650  000054  002761                  MOV     U.CBN+1(R5),CURBN+1     ; MOVE LBN TO CALCULATION AREA
016101  114007                        3$:     CLR     R0              ; TELL CALC ROUTINE TO SET UP PARAMETERS
016102                                          CALL    CALC            ; CALCULATE CYL AND GROUP
016102  020000  002604                          ^020000,CALC
016104  104203  002764                          MOV     #CYL,R3         ; R3 POINTS TO CALCULATED CYLINDER
016106  104657  000047                          MOV     U.RCOV(R5),R0   ; GET UNIT PARAMETERS
016110  102207  004000                          BIT     #SEKREQ,R0      ; SEE IF SEEK MANDATORY
016112                                          BNE     9$              ; IF SO, BRANCH
016112  050000  016131                          ^050000,9$
016114  104202  000064                          MOV     #U.CCYL,R2      ; R2 WILL POINT TO CURRENT CYL
016116  105052                                  ADD     R5,R2           ; R2 POINTS TO CURRENT CYLINDER
016117  104201  000003                          MOV     #3,R1           ; GET COUNT
016121  104237                        4$:     MOV     (R3)+,R0        ; GET WORD
016122  106227                                  CMP     (R2)+,R0        ; SEE IF CYL AND GROUP THE SAME
016123                                          BNE     9$              ; IF NOT, BRANCH
016123  050000  016131                          ^050000,9$
016125  117401                                  DEC     R1              ; DECREMENT COUNT
016126                                          BNE     4$              ; IF COUNT INCOMPLETE, BRANCH
016126  050000  016121                          ^050000,4$
016130  114003                                  CLR     R3              ; FLAG AS SEEK NOT NECESSARY
016131                                  9$:     RETURN                  ; RETURN TO SEEK
016131  000000  000000                          ^00,0
```

```
                                .SBTTL  ISUSEK - ISSUE SEEK COMMAND
016133                          ISUSEK:
                                ;
                                ;       ISUSEK ISSUES THE SEEK TO THE DRIVE
                                ;
016133  104300  002764  002433         MOV     CYL,INS+1       ; MOVE LOW CYL TO SEEK COMMAND
016136  104300  002765  002434         MOV     CYL+1,INS+2     ; SET LOWER BITS
016141  104300  002766  002435         MOV     GROUP,INS+3     ; MOVE GROUP TO SEEK COMMAND
016144  104203  002376                 MOV     #CR.SEK,R3      ; SET UP FOR TALK
016146                                 CALL    TALK            ; SEND SEEK
016146  020000  001670                 ^O20000,TALK
016150  115002                         TST     R2              ; SEE IF ERROR OCCURRED
016151                                 BNE     10$             ; IF SO, BRANCH
016151  050000  016213                 ^O50000,10$
016153  104207  000064                 MOV     #U.CCYL,R0      ; R0 WILL POINT TO CURRENT CYLINDER
016155  105057                         ADD     R5,R0           ; R0 POINTS TO CURRENT CYLINDER
016156  104201  000003                 MOV     #3,R1           ; MOVE THREE WORDS
016160                                 PUSH    <R0,R1>         ; SAVE POINTER AND COUNT
016160  100467                                                                 MOV R0,-(SP)
016161  100461                                                                 MOV R1,-(SP)
016162  104202  000067                 MOV     #U.LCYL,R2      ; R2 WILL POINT TO LAST CYLINDER
016164  105052                         ADD     R5,R2           ; R2 POINTS TO LAST CYLINDER
016165  104273                 1$:     MOV     (R0)+,R3        ; GET WORD
016166  100223                         MOV     R3,(R2)+        ; SAVE
016167  117401                         DEC     R1              ; DECREMENT COUNT
016170                                 BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
016170  050000  016165                 ^O50000,1$
016172                                 POP     <R2,R0>         ; RESTORE
016172  104262                                                                 MOV (SP)+,R2
016173  104267                                                                 MOV (SP)+,R0
016174  104201  002764         4$:     MOV     #CYL,R1         ; R2 POINTS TO NEW CYL
016176  104213                 2$:     MOV     (R1)+,R3        ; GET WORD
016177  100273                         MOV     R3,(R0)+        ; SAVE
016200  117402                         DEC     R2              ; DECREMENT COUNT
016201                                 BNE     2$              ; IF COUNT INCOMPLETE, BRANCH
016201  050000  016176                 ^O50000,2$
016203  104653  000046                 MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
016205  101203  002000                 BIS     #SEKINP,R3      ; MARK SEEK IN PROGRESS
016207  100653  000046                 MOV     R3,U.PARM(R5)   ; SAVE
016211                                 BR      ISUEXT          ; BRANCH (NOTE THAT R2 IS ZERO - NO ERRORS)
016211  000000  016216                 ^O00,ISUEXT
016213                         10$:    CERROR  5,#SER9         ; REPORT SECONDARY ERROR
016213  104200  005050  001602                                                 MOV     #SER9,HRQ.05
016216                         ISUEXT: RETURN                  ; RETURN TO CALLING PROGRAM
016216  000000  000000                 ^O00,0
```

```
********** OVERLAY MODULE SEKTST - SEE IF THE SEEK IS COMPLETE

                                        .SBTTL  ********** OVERLAY MODULE SEKTST - SEE IF THE SEEK IS COMPLETE
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;
                                        ;
                                        ;
016220                                  SEKTST:
                                        ;
                                        ;       SEKTST TESTS TO SEE IF READ/WRITE READY OR ATTENTION IS HIGH
                                        ;       R2 RETURNED AS ZERO IF SEEK WAS SUCESSFUL, -1 IF SEEK INCOMPLETE
                                        ;       POSITIVE NON-ZERO IF AN ERROR WAS DETECTED
                                        ;
                                        .ENABL  LSB
016220                                  CALL    RTDS            ; GET REAL TIME  DRIVE STATE
016220  020000  001417                  ^O020000,RTDS
016222  115002                         TST     R2              ; SEE IF ERROR OCCURRED
016223                                  BEQ     1$              ; IF NOT, BRANCH
016223  010000  016232                  ^O010000,1$
016225  104207  016220                  MOV     #SEKTST,R0      ; RETRY THIS MODULE
016227  104051                         MOV     R5,R1           ; DEFERRED CALL
016230                                  BR      STSEXT          ; EXIT
016230  000000  016633                  ^O00,STSEXT
016232                          1$:     CALL    ENABLE          ; ENABLE ERROR RECOVERY
016232  020000  005036                  ^O020000,ENABLE
016234  115001                         TST     R1              ; SEE IF R/W RDY ASSERTED
016235                                  BPL     STSERR          ; IF NOT READY, BRANCH
016235  030000  016422                  ^O030000,STSERR
016237                                  ASSUME  RWRDY,100000    ; ASSUME R/W RDY IS SIGN BIT
016237  104657  000046                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
016241  103207  002000                  BIC     #SEKINP,R0      ; SEEK NO LONGER IN PROGRESS
016243  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE PARAMETERS
016245  104657  000047                  MOV     U.RCOV(R5),R0   ; GET RECOVERY WORD
016247  103207  004000                  BIC     #SEKREQ,R0      ; SEEK NO LONGER REQUIRED
016251  100657  000047                  MOV     R0,U.RCOV(R5)   ; SAVE
016253  104647  000001                  MOV     S.SEEK(R4),R0   ; GET NUMBER OF SEEKS ISSUED
016255  117407                         DEC     R0              ; DECREMENT SEEK COUNT
016256                                  BNE     SEKCNT          ; IF NO REPORT NEEDED, BRANCH
016256  050000  016274                  ^O050000,SEKCNT
016260  104650  000063  001576          MOV     U.UNUM(R5),HRQ.01 ; GET STARTING SUBUNIT NUMBER
016263  105650  000050  001576          ADD     U.SUBU(R5),HRQ.01 ; ADD OFFSET
016266  104207  060010                  MOV     #T4SEEK,R0      ; MOVE SEEK REPORT NUMBER TO R0
016270                                  CALL    HOSTRQ          ; REPORT TO HOST
016270  020000  001543                  ^O020000,HOSTRQ
016272  104207  001750                  MOV     #1000.,R0       ; SET UP FOR NEW COUNT
                                        .DSABL  LSB
016274  100647  000001          SEKCNT: MOV     R0,S.SEEK(R4)   ; SAVE COUNT
016276  114002                         CLR     R2              ; NO ERRORS
016277  104653  000007                  MOV     U.SRTY(R5),R3   ; GET RETRY COUNT
016301                                  BEQ     2$              ; IF NO RETRIES, BRANCH
016301  010000  016405                  ^O010000,2$
016303                                  REPSFT  SOFT            ; REPORT SOFT ERROR
016303  104200  000001  001577                                          MOV     #1,HRQ.02
016306  114000  001600                                                  CLR     HRQ.03
016310  114000  001601                                                  CLR     HRQ.04
016312  100467                                                                  MOV R0,-(SP)
```

********** OVERLAY MODULE SEKTST - SEE IF THE SEEK IS COMPLETE

```
        016313  104657  000063                                          MOV     U.UNUM(R5),R0
        016315  105657  000050                                          ADD     U.SUBU(R5),R0
        016317  104070  001576                                          MOV     R0,HRQ.01
        016321  104207  060007                                          MOV     #T4SOFT,R0
        016323  020000  001543          ^O20000,HOSTRQ
        016325  104267                                                          MOV (SP)+,R0
    71  016326                          SOFTER  15,<R3,S.LETR(R4),U.CBN(R5),U.CBN+1(R5)>
        016326  104200  001503  001601                                  MOV     #ER15,HRQ.04
        016331  104030  001602                                          MOV     R3,HRQ.05
        016333  104640  000005  001603                                  MOV     S.LETR(R4),HRQ.06
        016336  104650  000053  001604                                  MOV     U.CBN(R5),HRQ.07
        016341  104650  000054  001605                                  MOV     U.CBN+1(R5),HRQ.08
        016344  104202  147657                                          MOV     #15!ERSOFT+4000.,R2
        016346  104020  001577                                          MOV     R2,HRQ.02
        016350  104200  016350  001576                                  MOV     #.,HRQ.01
        016353  104200  060013  001575                                  MOV     #ERRMES,HRQ.RQ
    72  016356                          ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5),U.RBN+1(R5),U.CGRP(R5),U.CCYL(R5)>
        016356  104650  000046  001606                                  MOV     U.PARM(R5),HRQ.09
        016361  104200  005472  001607                                  MOV     #RBNTXT,HRQ.10
        016364  104650  000055  001610                                  MOV     U.RBN(R5),HRQ.11
        016367  104650  000056  001611                                  MOV     U.RBN+1(R5),HRQ.12
        016372  104650  000066  001612                                  MOV     U.CGRP(R5),HRQ.13
        016375  104650  000064  001613                                  MOV     U.CCYL(R5),HRQ.14
    73  016400                          ERRORC  U.CCYL+1(R5)
        016400  104650  000065  001614                                  MOV     U.CCYL+1(R5),HRQ.15
    74  016403                          ENDERR  0
        016403  114000  003013                                          CLR     ERRPOS          ; CLEAR THE POSITION
    75  016405  104207  007746      2$: MOV     #BUILDP,R0      ; BUILD THE READ/WRITE CHAIN NEXT
    76  016407  104653  000047          MOV     U.RCOV(R5),R3   ; GET RECOVERY WORDS
    77  016411  102203  030000          BIT     #LEVUSD!NXTLEV,R3       ; SEE IF ERROR RECOVERY LEVELS IN USE
    78  016413                          BEQ     1$              ; IF NOT, BRANCH
        016413  010000  016417          ^O10000,1$
    79  016415  104207  014205          MOV     #NEWLEV,R0      ; ISSUE ERROR RECOVERY COMMAND NEXT
    80  016417  114001              1$: CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
    81  016420                          BR      STSEXT          ; BRANCH
        016420  000000  016633          ^O0,STSEXT
    82  016422  102201  000100  STSERR: BIT     #AVAIL,R1       ; SEE IF DRIVE TIMEOUT HAS EXPIRED
    83  016424                          BEQ     1$              ; IF NOT, BRANCH
        016424  010000  016440          ^O10000,1$
    84  016426  104653  000046          MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
    85  016430  103203  002000          BIC     #SEKINP,R3      ; MARK AS SEEK NOT IN PROGRESS
    86  016432  100653  000046          MOV     R3,U.PARM(R5)   ; SAVE
    87  016434  104207  015642          MOV     #SEEK,R0        ; SEEK AGAIN
    88  016436                          BR      STSEXT          ; EXIT
        016436  000000  016633          ^O0,STSEXT
    89  016440  102201  000002      1$: BIT     #ATTN,R1        ; SEE IF ATTENTION ASSERTED
    90  016442                          BEQ     2$              ; IF NOT, BRANCH
        016442  010000  016537          ^O10000,2$
    91  016444                          REPSFT  SOFT,,SEEK      ; REPORT SEEK AND SOFT ERROR
        016444  104200  00000*  001577                                  MOV     #1,HRQ.02
        016447  114000  0016C(                                          CLR     HRQ.03
        016451  104200  00000i  001601                                  MOV     #1,HRQ.04
        016454  100467                                                          MOV R0,-(SP)
        016455  104657  000063                                          MOV     U.UNUM(R5),R0
        016457  105657  000050                                          ADD     U.SUBU(R5),R0
        016461  104070  001576                                          MOV     R0,HRQ.01
        016463  104207  060007                                          MOV     #T4SOFT,R0
```

```
        016465  020000  001543                 ^020000,HOSTRQ
        016467  104267                                                    MOV (SP)+,R0
     92 016470                                  SOFTER  1,<U.LGRP(R5),U.LCYL(R5),U.LCYL+1(R5),U.CGRP(R5),U.CCYL(R5)>
        016470  104200  000000  001601                                    MOV     #ER1,HRQ.04
        016473  104650  000071  001602                                    MOV     U.LGRP(R5),HRQ.05
        016476  104650  000067  001603                                    MOV     U.LCYL(R5),HRQ.06
        016501  104650  000070  001604                                    MOV     U.LCYL+1(R5),HRQ.07
        016504  104650  000066  001605                                    MOV     U.CGRP(R5),HRQ.08
        016507  104650  000064  001606                                    MOV     U.CCYL(R5),HRQ.09
        016512  104202  147641                                           MOV     #1!ERSOFT+4000.,R2
        016514  104020  001577                                           MOV     R2,HRQ.02
        016516  104200  016516  001576                                    MOV     #.,HRQ.01
        016521  104200  060013  001575                                    MOV     #ERRMES,HRQ.RQ
     93 016524                                  ERRORC  U.CCYL+1(R5)
        016524  104650  000065  001607                                    MOV     U.CCYL+1(R5),HRQ.10
     94 016527                                  ENDERR                  ; REPORT ATTN ASSERTED DURING SEEK
        016527  104200  000051  001610                                    MOV     #SER22,HRQ.11
        016532  104200  000014  003013                                    MOV     #11+1,ERRPOS    ; SET THE POSITION
     95 016535                                  BR      STSEXT          ; EXIT
        016535  000000  016633                  ^00,STSEXT
     96 016537                          2$:     CALL    DSABLE
        016537  020000  005046                  ^020000,DSABLE
     97 016541  104051                          MOV     R5,R1           ; MAKE R1 NON-ZERO FOR DEFERRED CALL
     98                                 ;       NOTE:   R2 IS ZERO AT THIS POINT
     99 016542  104207  016220                  MOV     #SEKTST,R0      ; SEEK TEST IS NEXT MODULE
    100 016544  104653  000006                  MOV     U.TIML(R5),R3   ; GET TIMEOUT VALUE
    101 016546  107203  000001                  SUB     #1,R3           ; DECREMENT TIMEOUT VALUE
    102 016550  100653  000006                  MOV     R3,U.TIML(R5)   ; SAVE TIMEOUT VALUE
    103 016552                                  BCC     STSEXT          ; IF NON-ZERO BRANCH
        016552  040000  016633                  ^040000,STSEXT
    104 016554  104653  000005                  MOV     U.TIMH(R5),R3   ; GET HI ORDER TIMEOUT
    105 016556  107203  000001                  SUB     #1,R3           ; DECREMENT COUNT
    106 016560  100653  000005                  MOV     R3,U.TIMH(R5)   ; SAVE
    107 016562                                  BCC     STSEXT
        016562  040000  016633                  ^040000,STSEXT
    108 016564                                  SOFTER  3,<U.LGRP(R5),U.LCYL(R5),U.LCYL+1(R5)>
        016564  104200  000166  001601                                    MOV     #ER3,HRQ.04
        016567  104650  000071  001602                                    MOV     U.LGRP(R5),HRQ.05
        016572  104650  000067  001603                                    MOV     U.LCYL(R5),HRQ.06
        016575  104650  000070  001604                                    MOV     U.LCYL+1(R5),HRQ.07
        016600  104202  147643                                           MOV     #3!ERSOFT+4000.,R2
        016602  104020  001577                                           MOV     R2,HRQ.02
        016604  104200  016604  001576                                    MOV     #.,HRQ.01
        016607  104200  060013  001575                                    MOV     #ERRMES,HRQ.RQ
    109 016612                                  ERRORC  <U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
        016612  104650  000066  001605                                    MOV     U.CGRP(R5),HRQ.08
        016615  104650  000064  001606                                    MOV     U.CCYL(R5),HRQ.09
        016620  104650  000065  001607                                    MOV     U.CCYL+1(R5),HRQ.10
    110 016623                                  ENDERR
        016623  104200  000051  001610                                    MOV     #SER22,HRQ.11
        016626  104200  000014  003013                                    MOV     #11+1,ERRPOS    ; SET THE POSITION
    111 016631                                  CALL    GOSEEK
        016631  020000  005066                  ^020000,GOSEEK
    112 016633                         STSEXT: BR      JMPRET          ; RETURN TO SEQUENCER
        016633  000000  003746                  ^00,JMPRET
```

```
                                .SBTTL  ********** OVERLAY MODULE RECALB - RECALIBRATION
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                ;
016635                          RECALB:
                                ;
                                ;
                                ;
                                        .ENABL  LSB
016635  104203  002415                  MOV     #CR.INR,R3      ; POINT TO THE INITIATE RECALIBRATION COMMAND
016637                                  CALL    TALK            ; SEND-RECEIVE SDI COMMAND
016637  020000  001670                  ^O020000,TALK
016641  115002                          TST     R2              ; SEE IF ANY ERRORS OCCURRED
016642                                  BEQ     1$              ; IF NOT, BRANCH
016642  010000  016703                  ^O010000,1$
                                ;
                                ;       FROM HERE TO THE LABEL '5$' IS CODE THAT SEES IF THE HOST WAS
                                ;       HALTED LONG ENOUGH FOR THE DRIVE TO GO OFFLINE DURING THE RECEIVE
                                ;       LEVEL 2 OPERATION (NOTE THAT LONG TIMEOUTS TALK TO THE HOST <<DURING>>
                                ;       THE TIMEOUT
                                ;
016644  106202  147747                  CMP     #71.!ERSOFT+4000.,R2    ; SEE IF TIMEOUT OF RECEIVE
016646                                  BNE     5$              ; IF NOT, BRANCH
016646  050000  016676                  ^O050000,5$
016650                                  CALL    RTDS            ; GET STATE
016650  020000  001417                  ^O020000,RTDS
016652  115002                          TST     R2              ; SEE IF ERROR
016653                                  BEQ     4$              ; IF NOT, BRANCH
016653  010000  016661                  ^O010000,4$
016655                                  CALL    GORTRY          ; RETRY THIS MODULE
016655  0200C,  005056                  ^O020000,GORTRY
016657                                  BR      2$              ; EXIT
016657  000000  016725                  ^O00,2$
016661  102201  000100          4$:     BIT     #AVAIL,R1       ; SEE IF DRIVE IS AVAILABLE
016663                                  BEQ     6$              ; IF NOT, BRANCH
016663  010000  016675                  ^O010000,6$
016665  104303  002421                  MOV     CR.INP -2.EOF,R3        ; GET INR OFFSET
016667  105053                          ADD     R5,R3           ; POINT TO RECALIBRATE ERROR COUNT
016670  104131                          MOV     (R3),R1         ; GET COUNT
016671  117401                          DEC     R1              ; DECREMENT COUNT
016672  100131                          MOV     R1,(R3)         ; SAVE
016673                                  BR      3$              ;
016673  000000  016713                  ^O00,3$
016675  104052                  6$:     MOV     R5,R2           ; FLAG ERROR
016676                          5$:     CERROR  5,#SER7         ; REPORT SECONDARY ERROR
016676  104200  005010  001602                                                  MOV     #SER7,HRQ.05
016701                                  BR      2$              ; EXIT
016701  000000  016725                  ^O00,2$
016703  104653  000047          1$:     MOV     U.RCOV(R5),R3   ; GET RECOVERY PARAMETERS
016705  103203  002000                  BIC     #RCBREQ,R3      ; CLEAR RECALIBRATION REQUESTED FLAG
016707  101203  004000                  BIS     #SEKREQ,R3      ; MARK SEEK AS REQUIRED
016711  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
016713  114002                  3$:     CLR     R2              ; NO ERRORS
```

********** OVERLAY MODULE RECALB - RECALIBRATION

```
     72 016714  100652  000064                  MOV     R2,U.CCYL(R5)   ; ZERO CYLINDER
     73 016716  100652  000065                  MOV     R2,U.CCYL+1(R5)
     74 016720  100652  000066                  MOV     R2,U.CGRP(R5)
     75 016722  114001                          CLR     R1              ; IMMIDATE CALL
     76 016723  104207  015642                  MOV     #SEEK,R0        ; SEEK IS NEXT MODULE
     77 016725                          2$:     BR      JMPRET          ; RETURN TO SEQNCR
        016725  000000  003746                  ^O0,JMPRET
     78                                         .DSABL  LSB
```

```
                                .SBTTL  ********** OVERLAY MODULE DRPALL - DROP ALL SUBUNITS ON THIS DRIVE
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                ;
016727                          DRPALL:
                                ;
                                ;       DROP ALL SUBUNITS ON THIS DRIVE AND REPORT
                                ;
                                ;       NOTE:
                                ;               THIS IS A VERY UNSTRUCTURED PIECE OF CODE -- IT HAS SEVERAL
                                ;               JUMPS DIRECTLY INTO SEQNCR AT SEVERAL DIFFERENT PLACES.
                                ;
                                        .ENABL  LSB
016727                          PUSH    R4                      ; SAVE R4 (JUST IN CASE)
016727  100464                                                                  MOV R4,-(SP)
016730  104200  005545  001577  MOV     #MS2,HRQ.02             ; POINT TO MESSAGE
016733  104207  001601          MOV     #HRQ.04,R0              ; POINT TO WHERE TO PUT UNITS
016735  104053                  MOV     R5,R3                   ; GET POINTER TO UNIT DATABASE
016736  115403                  INC     R3                      ; POINT TO SUBUNIT POINTERS
016737                          ASSUME  U.SUBP,1
016737  114001                  CLR     R1                      ; CLEAR INDEX
016740  114004                  CLR     R4                      ; CLEAR NUMBER OF SUBUNITS COUNT
016741  104232          1$:     MOV     (R3)+,R2                ; GET POINTER
016742                          BMI     2$                      ; IF NO SUBUNIT, BRANCH
016742  070000  016753          ^070000,2$
016744  104652  000063          MOV     U.UNUM(R5),R2           ; GET BASE UNIT NUMBER
016746  105012                  ADD     R1,R2                   ; ADD OFFSET
016747  100272                  MOV     R2,(R0)+                ; MOVE TO OUTPUT BUFFER
016750  104020  001576          MOV     R2,HRQ.01               ; MOVE TO UNIT NUMBER
016752  115404                  INC     R4                      ; INC SUBUNIT COUNT
016753  115401          2$:     INC     R1                      ; INC COUNT
016754  106201  000003          CMP     #3,R1                   ; SEE IF EXPIRED
016756                          BCC     1$                      ; IF NOT, BRANCH
016756  040000  016741          ^040000,1$
016760  104647  017000          MOV     SER18F-1(R4),R0 ; POINT TO DRIVE LIST
016762  104070  001600          MOV     R0,HRQ.03               ; POINT TO CORRECT ERROR MESSAGE
016764  104207  060015          MOV     #MESSAG,R0              ; MESSAGE TO R0
016766                          CALL    HOSTRQ                  ; REPORT
016766  020000  001543          ^020000,HOSTRQ
016770  104651  000046          MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
016772  101201  100000          BIS     #DROP,R1                ; SET DROP BIT
016774  100651  000046          MOV     R1,U.PARM(R5)   ; SAVE
016776                          POP     R4
016776  104264                                                                  MOV (SP)+,R4
016777                          BR      NOSUB
016777  000000  004501          ^O0,NOSUB

017001  005432          SER18F: .WORD   SER18A
017002  005426                  .WORD   SER18B
017003  005422                  .WORD   SER18C
017004  005416                  .WORD   SER18D
                                .DSABL  LSB
017005                  BUFARA  =       .
```

```
                                        .SBTTL  ********** OVERLAY MODULE INSET - SET UP UNIT FOR INITIALZATION
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;
                                        ;
                                        ;
017005                                  INSET:
                                        ;
                                        ;       INSET WILL SET UP EACH UNIT BEFORE IT STARTS RUNNING
                                        ;
                                                .ENABL  LSB
017005  104657  000046                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
017007  103207  004000                  BIC     #NEWSUB,R0      ; NO LONGER NEW SUBUNITS
017011  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE
017013  114002                          CLR     R2              ; NO ERRORS
017014  100652  000064                  MOV     R2,U.CCYL(R5)   ; CLEAR CYL+GROUP
017016  100652  000065                  MOV     R2,U.CCYL+1(R5)
017020  100652  000066                  MOV     R2,U.CGRP(R5)
017022  114001                          CLR     R1              ; IMMIDATE CALL
017023  104207  017027                  MOV     #COMCHR,R0      ; COMMON CHARACTERISTICS NEXT MODULE CALLED
017025                                  BR      JMPRET          ; RETURN TO SEQUENCER
017025  000000  003746                  ^O0,JMPRET
                                        .DSABL  LSB
```

```
                                        .SBTTL  ********** OVERLAY MODULE COMCHR - SET UP COMMON CHARACTERISTICS
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;
                                        ;
                                        ;
017027                                  COMCHR:
                                        ;
                                        ;       GET COMMON CHARACTERISTICS AND SET UP THE ERROR RECOVERY LEVEL,
                                        ;       RETRIES, LONG TIMEOUT, AND SHORT TIMEOUT
                                        ;
                                        .ENABL  LSB
017027  104203  022606                  MOV     #CR.GCR,R3      ; POINT TO GET CHARACTERISTICS DATA BLOCK
017031                                  CALL    TALK            ; GET CHARACTERISTICS
017031  020000  001670                  ^O20000,TALK
017033  115002                          TST     R2              ; SEE IF ANY ERRORS OCCURRED
017034                                  BEQ     1$              ; IF NOT, BRANCH
017034  010000  017054                  ^O10000,1$
017036                                  CERROR  5,#SER8         ; REPORT SECONDARY ERROR
017036  104200  005027  001602                                          MOV     #SER8,HRQ.05
017041  103200  100000  001577          BIC     #C2DFTL,HRQ.02  ; CHANGE TO HARD ERROR
017044  104200  060014  001575          MOV     #ERRMC,HRQ.RQ   ; COUNT ERROR
017047  101200  000002  003006          BIS     #DIE,M.PARM     ; FLAG INITIALIZATION ERROR
017052                                  BR      2$              ; BRANCH TO EXIT
017052  000000  017164                  ^O00,2$
017054  104300  002446  022651  1$:     MOV     ST+DRVID,DSERNM ; SAVE DRIVE SERIAL NUMBER
017057  104300  002447  022652          MOV     ST+DRVID+1,DSERNM+1     ; SAVE DRIVE SERIAL NUMBER
017062  104300  002450  022653          MOV     ST+DRVID+2,DSERNM+2     ; SAVE DRIVE SERIAL NUMBER
017065  104307  002443                  MOV     ST+RETS,R0      ; GET NUMBER OF RETRIES ALLOWED
017067  110607                          ROR     R0              ; ROTATE TO CORRECT POSITION
017070  110607                          ROR     R0
017071  110607                          ROR     R0
017072  110607                          ROR     R0
017073  103207  177760                  BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
017075  100657  000030                  MOV     R0,U.RTRY(R5)   ; SAVE IN UNIT PARAMETERS
017077  104307  002444                  MOV     ST+ERLEV,R0     ; GET ERROR RECOVERY LEVELS
017101  103207  177400                  BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
017103  100657  000031                  MOV     R0,U.MLEV(R5)   ; SAVE IN UNIT PARAMETERS
017105  104307  002444                  MOV     ST+ECCRSH,R0    ; GET ECC THRESHOLD
017107  110707                          SWAB    R0              ; MOVE ECC THRESHOLD TO LOWER BYTE
017110  103207  177400                  BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
017112  100657  000032                  MOV     R0,U.ECCT(R5)   ; SAVE
017114  104307  002443                  MOV     ST+LONGTO,R0    ; GET LONG TIMEOUT
017116  103207  177760                  BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
017120                                  CALL    TO              ; CALCULATE TIMEOUT
017120  020000  017167                  ^O20000,TO
017122  100657  000034                  MOV     R0,U.SDIL(R5)   ; SAVE IN UNIT PARAMETERS
017124  104307  002442                  MOV     ST+SHRTTO,R0    ; GET SHORT TIMEOUT
017126  103207  177760                  BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
017130                                  CALL    TO              ; CALCULATE TIMEOUT
017130  020000  017167                  ^O20000,TO
017132  100657  000033                  MOV     R0,U.SDIS(R5)   ; SAVE SHORT TIMEOUT IN UNIT PARAMETERS
017134  114001                          CLR     R1              ; CLEAR LO SEEK TIMEOUT
017135  114002                          CLR     R2              ; CLEAR HI TIMEOUT
017136  105201  025762          4$:     ADD     #11250.,R1      ; ADD 9 SEC TIMEOUT TO LO ORDER TIMEOUT
```

```
                                        BCC     3$              ; IF NO CARRY, BRANCH
 70 017140                              ^040000,3$
    017140  040000  017143
                                        INC     R2              ; PROPOGATE CARRY
 71 017142  115402
 72 017143  117407              3$:     DEC     R0              ; DECREMENT TIMEOUT
                                        BNE     4$              ; IF UNEXPIRED, BRANCH
 73 017144                              ^050000,4$
    017144  050000  017136
                                        INC     R2              ; ROUND UP
 74 017146  115402
 75 017147  100652  000011             MOV     R2,U.MSTO(R5)   ; SAVE SEEK MASTER TIMEOUT
 76 017151  104307  002443             MOV     ST+RCTCPS,R0    ; GET NUMBER OF RCT COPIES
 77 017153  110707                      SWAB    R0              ; MOVE TO LOW WORD
 78 017154  103207  177760             BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
 79 017156  117407                      DEC     R0              ; ADJUST FOR TEST 4 INTERNALS
 80 017157  100657  000057             MOV     R0,U.COPY(R5)   ; SAVE
 81 017161  104207  017204             MOV     #SPINUP,R0      ; SPINUP NEXT MODULE
 82 017163  114002                      CLR     R2              ; NO ERRORS
 83 017164  114001              2$:     CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
 84 017165                              BR      JMPRET          ; RETURN TO SEQUNCER
    017165  000000  003746             ^00,JMPRET
 85                                     .DSABL  LSB
 86 017167                      TO:
 87                                     ;
 88                                     ;       CALCULATE THE TIMEOUT IN 9SEC INTERVALS (UDA SDI RECEIVE XFC TAKES
 89                                     ;       9 SEC)
 90                                     ;
 91 017167  104201  000001             MOV     #1,R1           ; SET UP LOG2 SHIFTER
 92 017171  105011              1$:     ADD     R1,R1           ; DOUBLE THE TIMEOUT VALUE
 93 017172  117407                      DEC     R0              ; DECREMENT COUNT
 94 017173                              BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
    017173  050000  017171             ^050000,1$
 95 017175  115407              2$:     INC     R0              ; INCREMENT 9 SEC COUNT
 96 017176  107201  000011             SUB     #9.,R1          ; SUBTRACT 9 SEC FROM TIMEOUT
 97 017200                              BPL     2$              ; IF MORE TIME TO GO, BRANCH
    017200  030000  017175             ^030000,2$
 98 017202                              RETURN                  ; RETURN TO CALLING PROGRAM
    017202  000000  000000             ^00,0
```

```
********** OVERLAY MODULE SPINUP - SPIN THE DRIVE UP

                                        .SBTTL  ********** OVERLAY MODULE SPINUP - SPIN THE DRIVE UP
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;
                                        ;
                                        ;
017204                                  SPINUP:
                                        ;
                                        ;       SPINUP WILL SPIN THE DRIV UP IF SPUN DOWN
                                        ;
                                        .ENABL  LSB
017204  104203  022620                  MOV     #CR.RUN,R3      ; POINT TO RUN COMMAND
017206                                  CALL    TALK            ; SEND COMMAND
017206  020000  001670                  ^020000,TALK
017210  115002                          TST     R2              ; SEE IF ERROR OCCURRED
017211                                  BNE     1$              ; IF SO, BRANCH
017211  050000  017217                  ^050000,1$
017213  104207  017236                  MOV     #SORT,R0        ; SORT NEXT MODULE
017215                                  BR      2$              ; EXIT
017215  000000  017233                  ^00,2$
017217                          1$:     CERROR  5,#SER6         ; REPORT SECONDARY ERROR
017217  104200  004770  001602                                                  MOV     #SER6,HRQ.05
017222  103200  100000  001577          BIC     #C2DFTL,HRQ.02  ; CHANGE TO HARD ERROR
017225  104200  060014  001575          MOV     #ERRMC,HRQ.RQ   ; COUNT ERROR
017230  101200  000002  003006          BIS     #DIE,M.PARM     ; FLAG INITIALIZATION ERROR
017233  114001                  2$:     CLR     R1              ; IMMIDATE CALL
017234                                  BR      JMPRET          ; RETURN TO SEQNCR
017234  000000  003746                  ^00,JMPRET
                                        .DSABL  LSB
```

```
                                .SBTTL  ********** OVERLAY MODULE SORT - SORT ALL CYLINDERS, BEGIN/SETS, AND BAD BLOCKS
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                ;
017236                          SORT:
                                ;
                                ;       SORT WILL SORT ALL BEGIN/END SETS, BAD BLOCKS AND TRACKS/GROUPS IN
                                ;       ASCENDING ORDER
                                ;
                                        .ENABL  LSB
017236                          PUSH    <R4,R5>         ; SAVE R4 AND R5
017236  100464                                                          MOV R4,-(SP)
017237  100465                                                          MOV R5,-(SP)
017240  115405                  INC     R5              ; POINT TO SUBUNIT POINTERS
017241                          ASSUME  U.SUBP,1
017241  104204  000004          MOV     #4,R4           ; MAXIMUM OF 4 SUBUNITS
017243  104257                  4$:     MOV     (R5)+,R0        ; R0 POINTS TO SUBUNIT DATABASE
017244                          BMI     3$              ; IF NO SUBUNIT, BRANCH
017244  070000  017267                  ^070000,3$
017246                          PUSH    <R4,R5>         ; SAVE R4 AND R5
017246  100464                                                          MOV R4,-(SP)
017247  100465                                                          MOV R5,-(SP)
017250  104171                  MOV     (R0),R1         ; GET SUBUNIT PARAMETERS
017251                          ASSUME  S.PARM,0
017251  102201  000040          BIT     #BEUSED,R1      ; SEE IF BEGIN/END SETS USED
017253                          BNE     1$              ; IF SO, BRANCH
017253  050000  017261                  ^050000,1$
017255                          CALL    SORTTG          ; SORT THE TRACKS/GROUPS
017255  020000  017461                  ^020000,SORTTG
017257                          BR      2$              ; BRANCH
017257  000000  017263                  ^00,2$
017261                          1$:     CALL    SORTBE          ; SORT THE BEGIN/END SETS
017261  020000  017302                  ^020000,SORTBE
017263                          2$:     CALL    SORTBB          ; SORT THE BAD BLOCKS
017263  020000  017403                  ^020000,SORTBB
017265                          POP     <R5,R4>         ; RESTORE R5, R4
017265  104265                                                          MOV (SP)+,R5
017266  104264                                                          MOV (SP)+,R4
017267  117404                  3$:     DEC     R4              ; DECREMENT COUNT
017270                          BNE     4$              ; IF INCOMPLETE, BRANCH
017270  050000  017243                  ^050000,4$
017272                          POP     <R5,R4>         ; RESTORE R5, R4
017272  104265                                                          MOV (SP)+,R5
017273  104264                                                          MOV (SP)+,R4
017274  114001                  CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
017275  114002                  CLR     R2              ; NO ERRORS
017276  104207  017514          MOV     #SCHAR0,R0      ; SCHAR0 NEXT MODULE
017300                          BR      JMPRET
017300  000000  003746                  ^00,JMPRET
                                .DSABL  LSB
```

```
      1                                          .SBTTL  SORTBE - SORT THE BEGIN/END SETS IN ASCENDING ORDER
      2 017302                                   SORTBE:
      3                                          ;
      4                                          ;       SORT THE BEGIN/END SETS IN ASCENDING ORDER
      5                                          ;
      6 017302  104672  000016                          MOV     S.BESS+3(R0),R2 ; SEE IF ONLY ONE BEGIN END SET
      7 017304                                           BMI     4$              ; IF SO, EXIT (ALLREADY SORTED)
        017304  070000  017347                           ^070000,4$
      8 017306  104202  000013                          MOV     #S.BESS,R2      ; R2 WILL POINT TO START OF BEGIN/END SETS
      9 017310  105072                                   ADD     R0,R2           ; R2 POINTS TO BEGIN/END SETS
     10 017311  104023                           1$:     MOV     R2,R3           ; R3 WILL POINT TO NEXT BEGIN/END SET
     11 017312  105203  000004                   2$:     ADD     #4,R3           ; R3 POINTS TO BEGIN/END SETS
     12                                          ;
     13                                          ;       28 BIT COMPARE FOR BEGIN/END SET SORTING
     14                                          ;
     15 017314  104634  000003                          MOV     3(R3),R4        ; GET WORD THAT R3 POINTS TO
     16 017316  103204  170000                          BIC     #^CHBHINB,R4    ; STRIP OFF UNUSED BITS
     17 017320  104625  000003                          MOV     3(R2),R5        ; GET OTHER WORD TO COMPARE
     18 017322  106045                                  CMP     R4,R5           ; COMPARE
     19 017323                                          BNE     10$             ; IF DIFFERENCE IS FOUND, BRANCH
        017323  050000  017331                           ^050000,10$
     20 017325  104625  000002                          MOV     2(R2),R5        ; GET OTHER WORD TO COMPARE
     21 017327  106635  000002                          CMP     2(R3),R5        ; COMPARE
     22 017331                                  10$:    BCC     3$              ; IF R2->B/E <= R3->B/E THEN ALLREADY IN ORDER
        017331  040000  017335                           ^040000,3$
     23 017333                                          CALL    SWAPBE          ; SWAP TE BEGIN/END SETS
        017333  020000  017351                           ^020000,SWAPBE
     24 017335  104635  000003                  3$:     MOV     3(R3),R5        ; SEE IF R3->END-OF-LIST
     25 017337                                          BPL     2$              ; IF NOT, BRANCH
        017337  030000  017312                           ^030000,2$
     26 017341  105202  000004                          ADD     #4,R2           ; R2->NEXT BEGIN/END SET
     27 017343  104625  000003                          MOV     3(R2),R5        ; SEE IF R2->END-OF-LIST
     28 017345                                          BPL     1$              ; IF NOT, BRANCH
        017345  030000  017311                           ^030000,1$
     29 017347                                  4$:     RETURN                  ; RETURN TO COPYSU
        017347  000000  000000                           ^00,0
```

```
                                .SBTTL  SWAPBE - IF BEGIN/END SETS OUT OF ORDER, SWAP
017351                          SWAPBE:
                                ;
                                ;       SWAP THE BEGIN/END SET POINTED TO BY R2 WITH THE BEGIN/END SET
                                ;       POINTED TO BY R3, KEEPING THE END-OF-LIST FLAG INTACT
                                ;
017351                                  PUSH    <R1,R2,R3>      ; SAVE POINTERS
017351  100461                                                          MOV R1,-(SP)
017352  100462                                                          MOV R2,-(SP)
017353  100463                                                          MOV R3,-(SP)
017354  104201  000003                 MOV     #3,R1           ; SET UP LOOP COUNT
017356  104124                  1$:     MOV     (R2),R4         ; GET WORD FROM SET
017357  104135                          MOV     (R3),R5         ; GET WORD FROM OTHER SET
017360  100234                          MOV     R4,(R3)+        ; SWAP WORD
017361  100225                          MOV     R5,(R2)+        ; SWAP WORD
017362  117401                          DEC     R1              ; DECREMENT COUNT
017363                                  BNE     1$              ; LOOP IF INCOMPLETE
017363  050000  017356                 ^O050000,1$
017365  104124                          MOV     (R2),R4         ; GET WORD FROM SET
017366  104135                          MOV     (R3),R5         ; GET WORD FROM OTHER SET
017367  104051                          MOV     R5,R1           ; MOVE TO TEMP STORAGE
017370  103205  177400                 BIC     #HIBYTE,R5      ; STRIP OFF END-OF-LIST FLAG, IF ANY
017372  107051                          SUB     R5,R1           ; R1 CONTAINS END-OF-LIST FLAG, IF ANY
017373  101014                          BIS     R1,R4           ; R4 NOW HAS END-OF-LIST FLAG, IF ANY
017374  100134                          MOV     R4,(R3)         ; SWAP WORD (AND EOL FLAG, IF ANY)
017375  100125                          MOV     R5,(R2)         ; SWAP WORD
017376                                  POP     <R3,R2,R1>      ; RESTORE THE REGISTERS
017376  104263                                                          MOV (SP)+,R3
017377  104262                                                          MOV (SP)+,R2
017400  104261                                                          MOV (SP)+,R1
017401                                  RETURN                  ; RETURN TO SORTBE
017401  000000  000000                 ^O00,0
```

SORTBB - SORT THE BAD BLOCKS IN ASCENDING ORDER

```
                                        .SBTTL  SORTBB - SORT THE BAD BLOCKS IN ASCENDING ORDER
017403                                  SORTBB:
                                        ;
                                        ;       SORT THE BAD BLOCKS IN ASCENDING ORDER
                                        ;
017403  104672  000012                  MOV     S.BADP(R0),R2   ; GET THE BAD BLOCK POINTER
017405                                  BEQ     4$              ; IF NO BAD BLOCKS, BRANCH
017405  010000  017457                  ^010000,4$
017407  104625  000001                  MOV     1(R2),R5        ; SEE IF ONLY ONE BAD BLOCK
017411                                  BMI     4$              ; IF SO BRANCH (ALLREADY SORTED)
017411  070000  017457                  ^070000,4$
017413  104023                  1$:     MOV     R2,R3           ; R3 WILL POINT AT NEXT BAD BLOCK
017414  105203  000002          2$:     ADD     #2,R3           ; R3 POINTS TO NEXT BAD BLOCK
017416                                  CALL    CBB2            ; COMPARE THE TWO BLOCKS
017416  020000  022773                  ^020000,CBB2
017420                                  BCC     3$              ; IF IN ASCENDING ORDER, BRANCH
017420  040000  017445                  ^040000,3$
                                        .SBTTL  SWAPBB - IF BAD BLOCKS OUT OF ORDER, SWAP
                                        ;SWAPBB
                                        ;
                                        ;       SWAP THE BAD BLOCKS, RETAINING THE END-OF-LIST POINTER AT THE END
                                        ;
017422                                  PUSH    R1              ; SAVE R1
017422  100461                                                                  MOV R1,-(SP)
017423  104124                          MOV     (R2),R4         ; GET LO ORDER BAD BLOCK OF 1ST SET
017424  104135                          MOV     (R3),R5         ; GET LO ORDER BAD BLOCK OF 2ND SET
017425  100134                          MOV     R4,(R3)         ; SWAP
017426  100125                          MOV     R5,(R2)         ; SWAP
017427  104624  000001                  MOV     1(R2),R4        ; GET HI ORDER BAD BLOCK OF 1ST SET
017431  104635  000001                  MOV     1(R3),R5        ; GET HI ORDER BAD BLOCK OF 2ND SET
017433  104051                          MOV     R5,R1           ; MOVE TO R1
017434  103205  177400                  BIC     #HIBYTE,R5      ; STRIP OFF END-OF-LIST FLAG, IF ANY
017436  107051                          SUB     R5,R1           ; R1 CONTAINS END-OF-LIST FLAG, IF ANY
017437  101014                          BIS     R1,R4           ; PUT END-OF-LIST FLAG IN R4, IF ANY
017440  100634  000001                  MOV     R4,1(R3)        ; SWAP
017442  100625  000001                  MOV     R5,1(R2)        ; SWAP
017444                                  POP     R1              ; RESTORE R1
017444  104261                                                                  MOV (SP)+,R1
017445  104635  000001          3$:     MOV     1(R3),R5        ; SEE IF END-OF-LIST
017447                                  BPL     2$              ; IF NOT, BRANCH
017447  030000  017414                  ^030000,2$
017451  105202  000002                  ADD     #2,R2           ; R2 POINTS TO NEXT BAD BLOCK
017453  104625  000001                  MOV     1(R2),R5        ; SEE IF END-OF-LIST
017455                                  BPL     1$              ; IF NOT, BRANCH
017455  030000  017413                  ^030000,1$
017457                          4$:     RETURN                  ; RETURN TO COPYSU
017457  000000  000000                  ^00,0
```

```
      1                                .SBTTL  SORTTG - SORT THE TRACK/GROUPS IN ASCENDING ORDER
      2 017461                         SORTTG:
      3                                ;
      4                                ;       SORTTG WILL SORT THE TRACKS OR GROUPS IN ASCENDING ORDER
      5                                ;
      6 017461  104201  000017                 MOV     #S.TGSS,R1      ; R1 WILL POINT TO TRACK/GROUP START
      7 017463  105071                         ADD     R0,R1           ; R1 POINTS TO TRACK/GROUP START
      8 017464  104115                 1$:     MOV     (R1),R5         ; GET START OF LIST
      9 017465                         .       BMI     4$              ; IF NEGATIVE, WHOLE LIST IS SORTED, EXIT
        017465  070000  017512                 ^070000,4$
     10 017467  104012                         MOV     R1,R2           ; R2 WILL POINT TO NEXT LIST MEMBER
     11 017470  115402                 2$:     INC     R2              ; R2 POINTS TO NEXT MEMBER
     12 017471  104125                         MOV     (R2),R5         ; GET NEXT MEMBER
     13 017472  104054                         MOV     R5,R4           ; COPY TO R4
     14 017473  103205  177400                 BIC     #HIBYTE,R5      ; CLEAR END-OF-LIST FLAG (IF ANY)
     15 017475  107054                         SUB     R5,R4           ; SAVE END-OF-LIST FLAG (IF ANY)
     16 017476  106115                         CMP     (R1),R5         ; SEE IF THE MEMBERS ARE IN ORDER
     17 017477                         .       BMI     3$              ; IF SO, BRANCH
        017477  070000  017504                 ^070000,3$
     18 017501  101114                         BIS     (R1),R4         ; COPY START OF LIST TO R4, RETAINING EOL FLAG
     19 017502  100115                         MOV     R5,(R1)         ; SWAP
     20 017503  100124                         MOV     R4,(R2)         ; SWAP
     21 017504  104125                 3$:     MOV     (R2),R5         ; GET MEMBER THAT SORT POINTER POINTS TO
     22 017505                         .       BPL     2$              ; IF NOT END-OF-LIST, BRANCH
        017505  030000  017470                 ^030000,2$
     23 017507  115401                         INC     R1              ; POINT TO NEXT START OF LIST
     24 017510                         .       BR      1$              ; LOOP
        017510  000000  017464                 ^00,1$
     25 017512                         4$:     RETURN                  ; RETURN TO CALLING PROGRAM
        017512  000000  000000                 ^00,0
```

```
                                .SBTTL  ********** OVERLAY MODULE SCHARO - SET UP SUBUNITS, CHECK THAT ALL PARAMETERS ARE WI
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
017514                          SCHARO:
                                ;
                                ;       INSCHR WILL GET THE SUBUNIT CHARACTERISTICS AND INITILIZE THE
                                ;       SUBUNIT PARAMETERS
                                ;
                                        .ENABL  LSB
017514                          PUSH    R4              ; SAVE R4 (SUBUNIT) POINTER
017514  100464                                                          MOV R4,-(SP)
017515  104041                  MOV     R4,R1           ; R1 POINTS TO SUBUNIT
017516  104657  000050          MOV     U.SUBU(R5),R0   ; GET SUBUNIT OFFSET
                                .SBTTL  SMASK  - CALCULATE THE SUBUNIT MASK
                                ;SMASK
                                ;
                                ;       SMASK TAKES THE UNIT OFFSET (0 - 3) IN R0 AND CHANGES IT TO THE
                                ;       SUBUNIT MASK (0001 - 1000)
                                ;
017520                          PUSH    R1              ; SAVE R1
017520  100461                                                          MOV R1,-(SP)
017521  104071                  MOV     R0,R1           ; MOVE R0 TO R1
017522  104207  000020          MOV     #20,R0          ; SUBUNIT 0 MASK
017524  115001                  TST     R1              ; SEE IF SUBUNIT MASK SHIFTED TO CORRECT POSITION
017525                  SMASKL: BEQ     SMASKX          ; IF SO, BRANCH
017525  010000  017533          ^010000,SMASKX
017527  110207                  ROL     R0              ; SHIFT MASK
017530  117401                  DEC     R1              ; DECREMENT COUNT
017531                          BR      SMASKL          ; BRANCH
017531  000000  017525          ^00,SMASKL
017533                  SMASKX: POP     R1              ; RESTORE R1
017533  104261                                                          MOV (SP)+,R1
017534  104070  022630          MOV     R0,SUBUNT       ; MOVE TO SDI BUFFER
017536  104203  022613          MOV     #CR.SCR,R3      ; POINT TO COMMAND
017540                          PUSH    R1              ; SAVE R1
017540  100461                                                          MOV R1,-(SP)
017541                          CALL    TALK            ; SDI EXCHANGE
017541  020000  001670          ^020000,TALK
017543                          POP     R1              ; RESTORE R1
017543  104261                                                          MOV (SP)+,R1
017544  115002                  TST     R2              ; SEE IF ERROR OCCURRED
017545                          BNE     13$             ; IF SO, BRANCH
017545  050000  020005          ^050000,13$
017547  100612  000010          MOV     R2,S.MEGR(R1)   ; ZERO MEGABIT COUNT
017551  100612  000011          MOV     R2,S.MEGW(R1)   ; ZERO MEGABIT COUNT
017553  104203  002442          MOV     #ST,R3          ; R3 POINTS TO SUBUNIT CHARACTERISTCS
017555  106205  037705          CMP     #FIRSTU,R5      ; IS R5 -> FIRST UNIT?
017557                          BEQ     1$              ; IF SO, BRANCH
017557  010000  017576          ^010000,1$
017561  104202  037705          MOV     #FIRSTU,R2      ; R2 -> FIRST UNIT
017563                          PUSH    <R1,R3,R4>      ; SAVE REGS
017563  100461                                                          MOV R1,-(SP)
```

```
        017564  100463                                                  MOV R3,-(SP)
        017565  100464                                                  MOV R4,-(SP)
    65  017566                          CALL    TRAV            ; TRAVERSE THE UNITS TO FIND IF SUBUNIT CHAR ARE EQUAL
        017566  020000  020027          ^020000,TRAV
    66  017570                          POP     <R4,R3,R1>      ; RESTORE REGS
        017570  104264                                                  MOV (SP)+,R4
        017571  104263                                                  MOV (SP)+,R3
        017572  104261                                                  MOV (SP)+,R1
    67  017573  115007                  TST     R0              ; WERE ANY CHARS EQUAL?
    68  017574                          BNE     3$              ; IF SO, BRANCH (R0 WILL BE NONE ZERO IF MATCH)
        017574  050000  017613          ^050000,3$
    69  017576  104207  000023  1$:     MOV     #19.,R0         ; MOVE WORD COUNT TO R0
    70  017600                          PUSH    R0              ; SAVE COUNT
        017600  100467                                                  MOV R0,-(SP)
    71  017601                          CALL    GETMEM          ; GET SOME MEMORY
        017601  020000  022765          ^020000,GETMEM
    72  017603  100617  000007          MOV     R0,S.SCHR(R1)   ; STORE POINTER TO IN SUBUNIT PARAMETERS
    73  017605                          POP     R2              ; RESTORE COUNT IN R2
        017605  104262                                                  MOV (SP)+,R2
    74  017606  104234  2$:     MOV     (R3)+,R4        ; MOVE ONE WORD OF S CHAR TO R0
    75  017607  100274                  MOV     R4,(R0)+        ; MOVE TO SUBUNIT CHAR AREA
    76  017610  117402                  DEC     R2              ; DECREMENT WORD COUNT
    77  017611                          BNE     2$              ; IF COUNT UNEXPIRED, BRANCH
        017611  050000  017606          ^050000,2$
    78  017613  104117  3$:     MOV     (R1),R0         ; GET SUBUNIT PARAMETERS
    79  017614                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
    80  017614  102207  020000          BIT     #DCYLS,R0       ; SEE IF DIAGNOSTIC CYLINDERS ARE USED
    81  017616                          BEQ     4$              ; IF NOT, BRANCH
        017616  010000  017630          ^010000,4$
    82  017620  104202  005521          MOV     #D$,R2          ; MOVE THE CHARACTER 'D' (TO MAKE DBN) TO R3
    83  017622  104303  002446          MOV     ST+RBNTRK,R3    ; GET RBN'S PER TRACK
    84  017624  103203  177600          BIC     #HIBYTE!200,R3  ; CLEAR UNUSED BITS
    85  017626                          BR      5$              ; BRANCH
        017626  000000  017633          ^00,5$
    86  017630  104202  005517  4$:     MOV     #L$,R2          ; MOVE THE CHARACTER 'L' (TO MAKE LBN) TO R3
    87  017632  114003                  CLR     R3              ; FOR LBN'S, NO RBN'S PER TRACK ADDED
    88  017633  100612  000005  5$:     MOV     R2,S.LETR(R1)   ; SAVE
    89  017635  105303  002453          ADD     ST+LBNTRK,R3    ; ADD LBNS PER TRACK (TO ZERO IF LBN AREA)
    90  017637  103203  177400          BIC     #HIBYTE,R3      ; CLEAR UNUSED BITS
    91  017641  104030  022637          MOV     R3,SECTRK       ; SAVE IN SECTORS PER TRACK
    92  017643  100613  000006          MOV     R3,S.TRKL(R1)   ; SAVE IN TRACK LENGTH
    93  017645                          CALL    COMPSC          ; COMPUTE SECTORS/GROUP AND SECTORS/CYL
        017645  020000  020657          ^020000,COMPSC
    94  017647                          CALL    CLCMAX          ; CALCULATE MAXIMUM DBN NUMBER (IN CASE NEEDED)
        017647  020000  020711          ^020000,CLCMAX
    95  017651  104117                  MOV     (R1),R0         ; GET SUBUNIT PARAMETERS
    96  017652                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
    97  017652  102207  000040          BIT     #BEUSED,R0      ; SEE IF BEGIN/END SETS ARE USED
    98  017654                          BEQ     6$              ; IF NOT, BRANCH
        017654  010000  017662          ^010000,6$
    99  017656  102207  000200          BIT     #ONLYCL,R0      ; SEE IF WE ALLREADY HAVE BEGIN/END SETS
   100  017660                          BEQ     7$              ; IF SO, BRANCH
        017660  010000  017667          ^010000,7$
   101  017662  6$:     CALL    CHKCYL          ; CONVERT THE CYLS TO BN'S
        017662  020000  020122          ^020000,CHKCYL
   102  017664  115002                  TST     R2              ; SEE IF AN ERROR OCCURRED
   103  017665                          BNE     9$              ; IF SO, BRANCH
```

```
                017665  050000  017716                  ^050000,9$
    104 017667                              7$:         CALL    CHKBES          ; CHECK THE BEGIN/END SETS FOR ERRORS
                017667  020000  020324                  ^020000,CHKBES
    105 017671  115002                                  TST     R2              ; SEE IF AN ERROR OCCURRED
    106 017672                                          BNE     9$              ; IF SO, BRANCH
                017672  050000  017716                  ^050000,9$
    107 017674  104302  002442                          MOV     ST+LBNCYL,R2    ; R2 CONTAINS LO LBN CYLS
    108 017676  105302  002463                          ADD     ST+XBNCYL,R2    ; ADD LO XBN CYLS TO R2
    109 017700  100612  000002                          MOV     R2,S.SDCL(R1)   ; MOVE TO LO STARTING DIAG CYL
    110 017702  104302  002443                          MOV     ST+LBNCYL+1,R2  ; GET HI ORDER LBN CYLS
    111 017704                                          BCC     8$              ; IF NO CARRY, BRANCH
                017704  040000  017707                  ^040000,8$
    112 017706  115402                                  INC     R2              ; PROPOGATE CARRY
    113 017707  100612  000003              8$:         MOV     R2,S.SDCL+1(R1) ; STORE HI STARTING DIAG CYL
    114 017711  114002                                  CLR     R2              ; NO ERRORS
    115 017712  104207  020761                          MOV     #SCHAR1,R0      ; SCHAR1 IS NEXT MODULE
    116 017714                                          BR      15$             ; EXIT
                017714  000000  020023                  ^00,15$
    117 017716  104650  000063  001600  9$:             MOV     U.UNUM(R5),HRQ.03       ; GET STARTING UNIT NUMBER
    118 017721  105650  000050  001600                  ADD     U.SUBU(R5),HRQ.03       ; ADD OFFSET
    119 017724  104307  001575                          MOV     HRQ.RQ,R0       ; SET UP TO REPORT ERROR
    120 017726                                          CALL    HOSTRQ          ; REPORT
                017726  020000  001543                  ^020000,HOSTRQ
    121 017730  101200  000002  003006                  BIS     #DIE,M.PARM     ; FLAG INITIALIZATION ERROR
    122 017733  104201  000001                          MOV     #U.SUBP,R1      ; R1 WILL POINT AT SUBUNIT POINTERS
    123 017735  105051                                  ADD     R5,R1           ; R1 POINTS AT SUBUNIT POINTERS
    124 017736  105651  000050                          ADD     U.SUBU(R5),R1   ; R1 POINTS AT POINTER TO SUBUNIT BEING HANDLED
    125 017740  104117                                  MOV     (R1),R0         ; R0 POINTS TO SUBUNIT BEGING HANDLED
    126 017741  104203  100000                          MOV     #DROP,R3        ; MOVE DROP FLAG TO R3
    127 017743  100173                                  MOV     R3,(R0)         ; DROP THIS SUBUNIT
    128 017744                                          ASSUME  S.PARM,0
    129 017744  114002                                  CLR     R2              ; NO ERRORS
    130 017745  104653  000050              10$:        MOV     U.SUBU(R5),R3   ; GET SUBUNIT OFFSET
    131 017747  115403                                  INC     R3              ; NEXT SUBUNIT
    132 017750  100653  000050                          MOV     R3,U.SUBU(R5)   ; SAVE
    133 017752  106203  000003                          CMP     #3,R3           ; SEE IF VALID OFFSET
    134 017754                                          BCS     11$             ; IF NOT, BRANCH
                017754  040000  017760                  ^040000,.+3
                017756  000000  017773                  ^00,11$
    135 017760  105203  000001                          ADD     #U.SUBP,R3      ; WILL POINT TO NEXT SUBUNIT POINTER
    136 017762  105053                                  ADD     R5,R3           ; ADD POINTER TO UNIT DATABASE
    137 017763  104133                                  MOV     (R3),R3         ; GET POINTER
    138 017764                                          BMI     10$             ; IF NO SUBUNIT, BRANCH
                017764  070000  017745                  ^070000,10$
    139 017766  104131                                  MOV     (R3),R1         ; GET SUBUNIT PARAMETERS
    140 017767                                          ASSUME  S.PARM,0
    141 017767                                          BMI     10$             ; IF DROPPED, BRANCH
                017767  070000  017745                  ^070000,10$
    142 017771                                          ASSUME  DROP,100000
    143 017771                                          BR      12$             ; EXIT
                017771  000000  017776                  ^00,12$
    144 017773  114003                      11$:        CLR     R3              ; NO SUBUNITS
    145 017774  100653  000050                          MOV     R3,U.SUBU(R5)   ; SAVE
    146 017776  115003                      12$:        TST     R3              ; ANY?
    147 017777                                          BNE     14$             ; IF SO, BRANCH
                017777  050000  020021                  ^050000,14$
    148 020001  104207  021707                          MOV     #GETSER,R0      ; GETSER NEXT MODULE
```

```
149 020003                                      BR      15$             ; EXIT
    020003  000000  020023                      ^O0,15$
150 020005                              13$:    CERROR  5,#SER5         ; REPORT SECONDARY ERROR
    020005  104200  004747  00°502                                                MOV     #SER5,HRQ.05
151 020010  103200  100000  0C.577              BIC     #C2DFTL,HRQ.02  ; CHANGE ERROR TO HARD ERROR
152 020013  104200  060014  001575              MOV     #ERRMC,HRQ.RQ   ; COUNT ERROR
153 020016  101200  000002  003006              BIS     #DIE,M.PARM     ; FLAG INITIALIZATION ERROR
154 020C21  104207  017514              14$:    MOV     #SCHAR0,R0      ; THIS MODULE IS NEXT
155 020023  114001                      15$:    CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
156 020024                                      POP     R4              ; RESTORE R4
    020024  104264                                                                  MOV (SP)+,R4
157 020025                                      BR      JMPRET          ; RETURN TO SEQNCR
    020025  000000  003746                      ^O0,JMPRET
158                                             .DSABL  LSB
```

```
                                .SBTTL  TRAV - TRAVERSE THOUGH THE UNITS TO FIND SUBUNIT CHARS THAT MATCH
020027                  TRAV:   PUSH    R2              ; SAVE R2 -> UNIT
020027  100462                                                            MOV R2,-(SP)
020030                          CALL    SUBTRV          ; GO TRAVERSE SUBUNITS
020030  020000  020044          ^020000,SUBTRV
020032                          POP     R2              ; RESTORE R2
020032  104262                                                            MOV (SP)+,R2
020033  115007                  TST     R0              ; FOUND A MATCH
020034                          BNE     1$              ; IF SO, BRANCH
020034  050000  020042          ^050000,1$
020036  104122                  MOV     (R2),R2         ; ELSE, CHECK NEXT UNIT
020037  106052                  CMP     R5,R2           ; ARE WE POINTING TO SAME UNIT?
020040                          BNE     TRAV            ; IF NOT, CONTINUE
020040  050000  020027          ^050000,TRAV
020042                  1$:     RETURN                  ; ELSE, EXIT
020042  000000  000000          ^O0,0

                                .SBTTL  SUBTRV - TRAVERSE THROUGH SUBUNITS TO FIND SUBUNIT CHARS THAT MATCH
020044  104204  000004  SUBTRV: MOV     #4,R4           ; R1 IS COUNTER TO END
020046  115402                  INC     R2              ; R2 -> SUBUNIT PARAMETER POINTER
020047                          ASSUME  U.SUBP,1
020047  104223          1$:     MOV     (R2)+,R3        ; R3 HAS ADDRESS OF SUBUNIT PARAMETER
020050                          BMI     2$              ; IF MINUS, NOT A GOOD SUBUNIT
020050  070000  020065          ^070000,2$
020052  104637  000007          MOV     S.SCHR(R3),R0   ; R0 -> SUBUNIT CHAR
020054                          PUSH    <R4,R2>
020054  100464                                                            MOV R4,-(SP)
020055  100462                                                            MOV R2,-(SP)
020056                          CALL    SCHRCP          ; CHECK SUBUNIT CHARACTERISTICS
020056  020000  020072          ^020000,SCHRCP
020060                          POP     <R2,R4>
020060  104262                                                            MOV (SP)+,R2
020061  104264                                                            MOV (SP)+,R4
020062  115007                  TST     R0              ; FIND A MATCH?
020063                          BNE     3$              ; IF SO, EXIT
020063  050000  020070          ^050000,3$
020065  117404          2$:     DEC     R4              ; ELSE, ARE ALL SUBUNITS DONE FOR THIS UNIT?
020066                          BNE     1$              ; IF NOT, BRANCH
020066  050000  020047          ^050000,1$
020070                  3$:     RETURN
020070  000000  000000          ^O0,0

                                .SBTTL  SCHRCP - SUBUNIT CHARACTERISTICS COMPARE
                                ; *** R0 = 0 IF NO MATCH OR R0 -> SUBUNIT CHAR IF THERE IS A MATCH
020072                  SCHRCP: PUSH    <R0,R1>
020072  100467                                                            MOV R0,-(SP)
020073  100461                                                            MOV R1,-(SP)
020074  104204  002442          MOV     #ST,R4          ; R4 -> CHARACTERISTICS
020076  104201  000023          MOV     #19.,R1         ; R1 = # OF WORDS TO COMPARE
020100  104242          1$:     MOV     (R4)+,R2        ; COMPARE CHARACTERISTICS
020101  106272                  CMP     (R0)+,R2        ; EQUAL?
020102                          BNE     2$              ; IF NOT, EXIT
020102  050000  020115          ^050000,2$
020104  117401                  DEC     R1              ; ELSE, CHECK NEXT WORD
020105                          BNE     1$              ; IF NOT DONE WITH COMPARISON, BRANCH
020105  050000  020100          ^050000,1$
020107                          POP     <R1,R0>         ; RESTORE REGS
```

```
        020107  104261                                                                  MOV (SP)+,R1
        020110  104267                                                                  MOV (SP)+,R0
     39 020111  100617  000007                MOV     R0,S.SCHR(R1)   ; AND STORE POINT TO CHAR INTO SUBUNIT PARAM
     40 020113                                BR      3$              ; AND EXIT WITH MATCH
        020113  000000  020120                ^O0,3$
     41 020115                        2$:     POP     <R1,R0>
        020115  104261                                                                  MOV (SP)+,R1
        020116  104267                                                                  MOV (SP)+,R0
     42 020117  114007                        CLR     R0              ; R0 = 0 (NO MATCH)
     43 020120                        3$:     RETURN
        020120  000000  000000                ^O0,0
```

```
                                        .SBTTL  CHKCYL - CHECK VALIDITY OF STARTING AND ENDING CYLS, CONVERT TO BEGIN/END SET
020122                                  CHKCYL:
                                        ;
                                        ;       CALCULATE THE STARTING AND ENDING BN'S FROM THE GIVEN STARTING
                                        ;       AND ENDING CYLS
                                        ;
020122                                          PUSH    R5              ; SAVE R4, R5
020122  100465                                                                 MOV R5,-(SP)
020123  104614  000015                          MOV     S.BESS+2(R1),R4 ; R4 HAS LO STARTING CYL
020125  104615  000016                          MOV     S.BESS+3(R1),R5 ; R5 HAS HI STARTING CYL
020127  103205  100000                          BIC     #100000,R5      ; CLEAR EOL FLAG (IF ANY)
020131                                          CALL    CYLBN           ; CALCULATE STARTING BN ON THAT CYL
020131  020000  020241                          ^020000,CYLBN
020133  115002                                  TST     R2              ; SEE IF ANY ERROR OCCURRED
020134                                          BNE     10$             ; IF SO, BRANCH
020134  050000  020236                          ^050000,10$
020136  100617  000015                          MOV     R0,S.BESS+2(R1) ; SAVE LO ORDER STARTING BN
020140  101203  100000                          BIS     #100000,R3      ; SET END-OF-LIST FLAG
020142  100613  000016                          MOV     R3,S.BESS+3(R1) ; SAVE HI ORDER STARTING BN
020144  104615  000014                          MOV     S.BESS+1(R1),R5 ; GET HI ORDER ENDING CYL
020146  106205  177777                          CMP     #-1,R5          ; SEE IF SHOULD DEFAULT TO HIGHEST LBN
020150                                          BEQ     4$              ; IF SO, BRANCH
020150  010000  020201                          ^010000,4$
020152  104614  000013                          MOV     S.BESS(R1),R4   ; GET LO ORDER ENDING CYL
020154  105204  000001                          ADD     #1,R4           ; FIND STARTING BN OF NEXT CYL
020156                                          BCC     2$              ; IF NO CARRY, BRANCH
020156  040000  020161                          ^040000,2$
020160  115405                                  INC     R5              ; PROPOGATE CARRY
020161                                  2$:     CALL    CYLBN           ; CACULATE STARTING BN OF NEXT CYL
020161  020000  020241                          ^020000,CYLBN
020163  115002                                  TST     R2              ; SEE IF ANY ERRORS
020164                                          BNE     10$             ; IF SO, BRANCH
020164  050000  020236                          ^050000,10$
020166  107207  000001                          SUB     #1,R0           ; GET LAST BN OF PREVIOUS CYL
020170                                          BCC     3$              ; IF NO CARRY, BRANCH
020170  040000  020173                          ^040000,3$
020172  117403                                  DEC     R3              ; PROPOGATE CARRY
020173  100617  000013                  3$:     MOV     R0,S.BESS(R1)   ; SAVE LO ORDER ENDING BN
020175  100613  000014                          MOV     R3,S.BESS+1(R1) ; SAVE HI ORDER ENDING BN
020177                                          BR      9$              ; EXIT
020177  000000  020235                          ^00,9$
020201  104117                          4$:     MOV     (R1),R0         ; GET SUBUNIT PARAMETERS
020202                                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
020202  102207  020000                          BIT     #DCYLS,R0       ; SEE IF DIAGNOSTIC CYLINDERS USED
020204                                          BNE     6$              ; IF SO, BRANCH
020204  050000  020225                          ^050000,6$
020206  104307  002454                          MOV     ST+LBNHST,R0    ; GET LO ORDER ENDING LBN
020210  107207  000001                          SUB     #1,R0           ; LAST LBN IN LBN AREA
020212  100617  000013                          MOV     R0,S.BESS(R1)   ; SAVE LO ORDER
020214  104307  002455                          MOV     ST+LBNHST+1,R0  ; GET HI ORDER
020216                                          BCC     5$              ; IF NO CARRY, BRANCH
020216  040000  020221                          ^040000,5$
020220  117407                                  DEC     R0              ; PROPOGATE CARRY
020221  100617  000014                  5$:     MOV     R0,S.BESS+1(R1) ; SAVE HI ORDER
020223                                          BR      9$              ; EXIT
020223  000000  020235                          ^00,9$
020225  104307  022635                  6$:     MOV     MAXDBN,R0       ; GET LO ORDER MAX DBN
```

```
     46 020227  100617  000013                 MOV     R0,S.BESS(R1)   ; SAVE
     47 020231  104307  022636                 MOV     MAXDBN+1,R0     ; GET HI ORDER
     48 020233  100617  000014                 MOV     R0,S.BESS+1(R1) ; SAVE
     49 020235  114002                  9$:    CLR     R2              ; NO ERRORS
     50 020236                          10$:   POP     R5              ; RESTORE
        020236  104265                                                          MOV (SP)+,R5
     51 020237                                 RETURN
        020237  000000  000000                 ^O0,0
```

```
                                        .SBTTL  CYLBN  - GIVEN A CYLINDER, CALCULATE STARTING BN ON THAT CYLINDER
020241                                  CYLBN:
                                        ;
                                        ;       TAKING THE CYL GIVEN IN R4,R5 CALCULATE THE FIRST BN ON THAT
                                        ;       CYL AND RETURN IN R2,R3.  IF BN EXCEEDS 28 BITS, REPORT
                                        ;       ERROR
                                        ;
020241  114007                          1$:     CLR     R0              ; CLEAR THE PRODUCT AREA
020242  114003                                  CLR     R3              ; CLEAR THE PRODUCT AREA
020243  115004                                  TST     R4              ; SEE IF LO ORDER IS ZERO
020244                                          BNE     1$              ; IF NOT, BRANCH
020244  050000  020251                          ^O50000,1$
020246  115005                                  TST     R5              ; SEE IF HI ORDER IS ZERO
020247                                          BEQ     5$              ; IF SO, BRANCH AND EXIT
020247  010000  020321                          ^O10000,5$
020251  107204  000001                  1$:     SUB     #1,R4           ; ADJUST COUNT
020253                                          BCC     2$              ; IF NO BORROW, BRANCH
020253  040000  020256                          ^O40000,2$
020255  117405                                  DEC     R5              ; PROPOGATE BORROW
020256  105307  022641                  2$:     ADD     SECCYL,R0       ; ADD LO SECTORS/CYL TO LO ORDER PROD
020260                                          BCC     3$              ; IF NO CARRY, BRANCH
020260  040000  020263                          ^O40000,3$
020262  115403                                  INC     R3              ; PROPOGATE CARRY
020263  106203  007777                  3$:     CMP     #HBHINB,R3      ; SEE IF HI ORDER TOO BIG
020265                                          BCC     4$              ; IF NOT, BRANCH
020265  040000  020311                          ^O40000,4$
020267                                          DEVFTL  57,#SER20       ; REPORT LBN OVERFLOW
020267  104200  003644  001601                                                  MOV     #ER57,HRQ.04
020272  104200  004047  001602                                                  MOV     #SER20,HRQ.05
020275  104202  047731                                                          MOV     #57!FTLDEV+4000.,R2
020277  104020  001577                                                          MOV     R2,HRQ.02
020301  104200  020301  001576                                                  MOV     #.,HRQ.01
020304  104200  060014  001575                                                  MOV     #ERRMC,HRQ.RQ
020307                                          BR      6$              ; EXIT
020307  000000  020322                          ^O00,6$
020311  107204  000001                  4$:     SUB     #1,R4           ; DECREMENT LO ORDER COUNT
020313                                          BCC     2$              ; IF INCOMPLETE, BRANCH
020313  040000  020256                          ^O40000,2$
020315  107205  000001                          SUB     #1,R5           ; DECREMENT HI ORDER COUNT
020317                                          BCC     2$              ; IF INCOMPLETE, BRANCH
020317  040000  020256                          ^O40000,2$
020321  114002                          5$:     CLR     R2              ; NO ERRORS
020322                                  6$:     RETURN                  ; RETURN TO CALLING PROGRAM
020322  000000  000000                          ^O00,0
```

```
                                        .SBTTL  CHKBES - CHECK THE BEGIN/END SETS FOR ERRORS
020324                                  CHKBES:
                                        ;
                                        ;       CHECK THAT THE BEGIN BLOCK <= END BLOCK IN EACH BEGIN/END SET,
                                        ;       THAT THE END BLOCK < THE BEGIN BLOCK OF THE NEXT BEGIN/END SET,
                                        ;       AND THAT THE LAST END BLOCK IS A VALID LBN OR DBN FOR THAT DRIVE.
                                        ;
020324                                          PUSH    R5
020324  100465                                                                            MOV R5,-(SP)
020325  104203  000013                          MOV     #S.BESS,R3      ; R3 WILL POINT TO ENDING BLOCK NUMBER
020327  105013                                  ADD     R1,R3           ; R3 POINTS TO ENDING BN
020330  104202  000015                          MOV     #S.BESS+2,R2    ; R2 WILL POINT TO STARTING BN
020332  105012                                  ADD     R1,R2           ; R2 POINTS TO STARTING BN
020333                                  1$:     CALL    CBB2            ; COMPARE
020333  020000  022773                          ^020000,CBB2
020335                                          BCC     2$              ; IF NO ERROR, BRANCH
020335  040000  020414                          ^040000,2$
020337  104112                                  MOV     (R1),R2         ; GET SUBUNIT PARAMETERS
020340                                          ASSUME  S.PARM,0
020340  102202  000040                          BIT     #BEUSED,R2      ; SEE IF BEGIN/END SETS USED
020342                                          BEQ     20$             ; IF NOT, BRANCH
020342  010000  020350                          ^010000,20$
020344  102202  000200                          BIT     #ONLYCL,R2      ; SEE IF ONLY THE CYLINDER IS SPECIFIED
020346                                          BEQ     21$             ; IF NOT, BRANCH
020346  010000  020372                          ^010000,21$
020350                                  20$:    DEVFTL  55,#SER20       ; REPORT BEGIN CYL > ENDING CYL
020350  104200  003552  001601                                                  MOV     #ER55,HRQ.04
020353  104200  004047  001602                                                  MOV     #SER20,HRQ.05
020356  104202  047727                                                          MOV     #55!FTLDEV+4000.,R2
020360  104020  001577                                                          MOV     R2,HRQ.02
020362  104200  020362  001576                                                  MOV     #.,HRQ.01
020365  104200  060014  001575                                                  MOV     #ERRMC,HRQ.RQ
020370                                          BR      7$              ; BRANCH TO ERROR EXIT
020370  000000  020654                          ^00,7$
020372                                  21$:    DEVFTL  50,#SER20       ; REPORT BEGIN > ENDING BN
020372  104200  003315  001601                                                  MOV     #ER50,HRQ.04
020375  104200  004047  001602                                                  MOV     #SER20,HRQ.05
020400  104202  047722                                                          MOV     #50!FTLDEV+4000.,R2
020402  104020  001577                                                          MOV     R2,HRQ.02
020404  104200  020404  001576                                                  MOV     #.,HRQ.01
020407  104200  060014  001575                                                  MOV     #ERRMC,HRQ.RQ
020412                                          BR      7$              ; BRANCH TO ERROR EXIT
020412  000000  020654                          ^00,7$
020414  104625  000001                  2$:     MOV     1(R2),R5        ; SEE IF THE END-OF-LIST HAS BEEN FOUND
020416                                          BMI     4$              ; IF SO, BRANCH
020416  070000  020456                          ^070000,4$
020420  105202  000004                          ADD     #4,R2           ; POINT TO NEXT BEGIN SET
020422                                          CALL    CBB2            ; COMPARE
020422  020000  022773                          ^020000,CBB2
020424                                          BCS     3$              ; BRANCH
020424  040000  020430                          ^040000,.+3
020426  000000  020452                          ^00,3$
020430                                          DEVFTL  51,#SER20       ; STARTING BN <= PREVIOUS ENDING BN
020430  104200  003362  001601                                                  MOV     #ER51,HRQ.04
020433  104200  004047  001602                                                  MOV     #SER20,HRQ.05
020436  104202  047723                                                          MOV     #51!FTLDEV+4000.,R2
020440  104020  001577                                                          MOV     R2,HRQ.02
```

```
                020442  104200  020442  001576                                          MOV     #.,HRQ.01
                020445  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
     31 020450                                          BR      7$              ; ERROR EXIT
                020450  000000  020654                  ^O00,7$
     32 020452  105203  000004          3$:     ADD     #4,R3           ; R3 NOW POINTS TO NEXT END SET
     33 020454                                  BR      1$              ; LOOP
                020454  000000  020333                  ^O00,1$
     34 020456  104112                  4$:     MOV     (R1),R2         ; GET SUBUNIT CHARACTERISTICS
     35 020457                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
     36 020457  102202  020000                  BIT     #DCYLS,R2       ; SEE IF DIAGNOSTIC CYLINDERS ARE IN USE
     37 020461                                  BNE     5$              ; IF SO, BRANCH
                020461  050000  020553                  ^O50000,5$
     38 020463  104202  002454                  MOV     #ST+LBNHST,R2   ; R2 POINTS TO LBNS IN HOST AREA
     39 020465                                  CALL    CBB2            ; COMPARE
                020465  020000  022773                  ^O20000,CBB2
     40 020467                                  BCS     6$              ; EXIT
                020467  040000  020473                  ^O40000,.+3
                020471  000000  020653                  ^O00,6$
     41 020473  104112                          MOV     (R1),R2         ; GET SUBUNIT CHARACTERISTICS
     42 020474                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
     43 020474  102202  000040                  BIT     #BEUSED,R2      ; SEE IF TRACKS/GROUPS IN USE
     44 020476                                  BEQ     10$             ; IF SO, BRANCH
                020476  010000  020504                  ^O10000,10$
     45 020500  102202  000200                  BIT     #ONLYCL,R2      ; SEE IF B/E SET COMPUTED BY CYLINDERS
     46 020502                                  BEQ     11$             ; IF NOT, BRANCH
                020502  010000  020514                  ^O10000,11$
     47 020504                          10$:    CERROR  6,#L$           ; LBN ERROR
                020504  104200  005517  001603                                          MOV     #L$,HRQ.06
     48 020507                                  CERROR  7,#L$           ; LBN ERROR
                020507  104200  005517  001604                                          MOV     #L$,HRQ.07
     49 020512                                  BR      14$             ; EXIT
                020512  000000  020601                  ^O00,14$
     50 020514  107200  000001  002454  11$:    SUB     #1,ST+LBNHST    ; DECREMENT LO LBN COUNT BY 1
     51 020517                                  BCC     9$              ; IF NO BORROW, BRANCH
                020517  040000  020523                  ^O40000,9$
     52 020521  117400  002455                  DEC     ST+LBNHST+1     ; PROPOGATE BORROW
     53 020523                          9$:     DEVFTL  52,<#SER20,ST+LBNHST,ST+LBNHST+1> ; LAST ENDING BN > MAX
                020523  104200  003405  001601                                          MOV     #ER52,HRQ.04
                020526  104200  004047  001602                                          MOV     #SER20,HRQ.05
                020531  104300  002454  001603                                          MOV     ST+LBNHST,HRQ.06
                020534  104300  002455  001604                                          MOV     ST+LBNHST+1,HRQ.07
                020537  104202  047724                                                  MOV     #52!FTLDEV+4000.,R2
                020541  104020  001577                                                  MOV     R2,HRQ.02
                020543  104200  020543  001576                                          MOV     #.,HRQ.01
                020546  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
     54 020551                                  BR      7$              ; ERROR EXIT
                020551  000000  020654                  ^O00,7$
     55 020553  104032                  5$:     MOV     R3,R2           ; R2 POINTS TO LAST END SET
     56 020554  104203  022635                  MOV     #MAXDBN,R3      ; R3 POINTS TO MAXIMUM DBN
     57 020556                                  CALL    CBB2            ; COMPARE
                020556  020000  022773                  ^O20000,CBB2
     58 020560                                  BCC     6$              ; EXIT
                020560  040000  020653                  ^O40000,6$
     59 020562  104112                          MOV     (R1),R2         ; GET SUBUNIT CHARACTERISTICS
     60 020563                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
     61 020563  102202  000040                  BIT     #BEUSED,R2      ; SEE IF TRACKS/GROUPS IN USE
     62 020565                                  BEQ     12$             ; IF SO, BRANCH
```

```
                020565  010000  020573                  ^010000,12$
     63 020567  102202  000200                  BIT     #ONLYCL,R2      ; SEE IF B/E SET COMPUTED BY CYLINDERS
     64 020571                                  BEQ     13$             ; IF NOT, BRANCH
        020571  010000  020623                  ^010000,13$
     65 020573                          12$:    CERROR  6,#D$           ; DBN ERROR
        020573  104200  005521  001603                                          MOV     #D$,HRQ.06
     66 020576                                  CERROR  7,#D$           ; DBN ERROR
        020576  104200  005521  001604                                          MOV     #D$,HRQ.07
     67 020601                          14$:    DEVFTL  62,#SER20       ; REPORT ERROR
        020601  104200  003765  001601                                          MOV     #ER62,HRQ.04
        020604  104200  004047  001602                                          MOV     #SER20,HRQ.05
        020607  104202  047736                                                 MOV     #62!FTLDEV+4000.,R2
        020611  104020  001577                                                 MOV     R2,HRQ.02
        020613  104200  020613  001576                                          MOV     #.,HRQ.01
        020616  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
     68 020621                                  BR      7$              ; EXIT
        020621  000000  020654                  ^00,7$
     69 020623                          13$:    DEVFTL  52,<#SER20,MAXDBN,MAXDBN+1> ; LAST ENDING BN > MAX
        020623  104200  003405  001601                                          MOV     #ER52,HRQ.04
        020626  104200  004047  001602                                          MOV     #SER20,HRQ.05
        020631  104300  022635  001603                                          MOV     MAXDBN,HRQ.06
        020634  104300  022636  001604                                          MOV     MAXDBN+1,HRQ.07
        020637  104202  047724                                                 MOV     #52!FTLDEV+4000.,R2
        020641  104020  001577                                                 MOV     R2,HRQ.02
        020643  104200  020643  001576                                          MOV     #.,HRQ.01
        020646  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
     70 020651                                  BR      7$              ; ERROR EXIT
        020651  000000  020654                  ^00,7$
     71 020653  114002                  6$:     CLR     R2              ; FLAG AS NO ERROR
     72 020654                          7$:     POP     R5              ; RESTORE REGISTER
        020654  104265                                                                 MOV (SP)+,R5
     73 020655                                  RETURN                  ; RETURN TO INSCHR
        020655  000000  000000                  ^00,0
```

```
                                        .SBTTL  COMPSC - COMPUTE SECTORS/GROUP AND SECTORS/CYLINDER
020657                                  COMPSC:
                                        ;
                                        ;       COMPUTE HOW MANY SECTORS/GROUP AND SECTORS/CYLINDER
                                        ;       GIVEN HOW MANY SECTORS/TRACK IN SECTRK
                                        ;
020657  114007                                  CLR     R0              ; CLEAR RUNNING TOTAL
020660  104302  002445                          MOV     ST+TRKGRP,R2    ; GET NUMBER OF TRACKS PER GROUP
020662  103202  177400                          BIC     #HIBYTE,R2      ; CLEAR UNUSED BITS
020664  105307  022637                1$:       ADD     SECTRK,R0       ; ADD NUMBER OF SECTORS/TRACK TO RUNNING TOTAL
020666  117402                                  DEC     R2              ; DECREMENT COUNT
020667                                          BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
020667  050000  020664                          ^050000,1$
020671  104070  022640                          MOV     R0,SECGRP       ; SAVE SECTORS/GROUP
020673  114007                                  CLR     R0              ; CLEAR RUNNING TOTAL
020674  104302  002444                          MOV     ST+GRPCYL,R2    ; GET GROUPS/CYLINDER
020676  103202  177400                          BIC     #HIBYTE,R2      ; CLEAR UNUSED BITS
020700  105307  022640                2$:       ADD     SECGRP,R0       ; ADD SECTORS/GROUP TO RUNNING TOTAL
020702  117402                                  DEC     R2              ; DECREMENT COUNT
020703                                          BNE     2$              ; IF COUNT INCOMPLETE, BRANCH
020703  050000  020700                          ^050000,2$
020705  104070  022641                          MOV     R0,SECCYL       ; STORE SECTORS/CYLINDER
020707                                          RETURN                  ; RETURN TO CALLING PROGRAM
020707  000000  000000                          ^00,0
```

```
                                .SBTTL  CLCMAX - CALCULATE THE MAXIMUM WRITEABLE DBN
020711                          CLCMAX:
                                ;
                                ;       CALCULATE THE MAXIMUM DBN NUMBER ALLOWED, AND PLACE IT IN MAXDBN
                                ;       FOR BEGIN/END SET GENERATION AND TESTING (BAD BLOCKS TOO)
                                ;
020711                                  PUSH    R1              ; SAVE REGISTER
020711  100461                                                                  MOV R1,-(SP)
020712  114007                          CLR     R0              ; CLEAR PRODUCT AREA
020713  114001                          CLR     R1              ; CLEAR PRODUCT AREA
020714  104302  002464                  MOV     ST+DBNCYL,R2    ; GET DIAG CYL COUNT
020716  110702                          SWAB    R2              ; MOVE TO LO ORDER BYTE
020717  103202  177400                  BIC     #HIBYTE,R2      ; CLEAR UNUSED BITS
020721  105307  022641          1$:     ADD     SECCYL,R0       ; ADD SECTORS/CYL TO TOTAL
020723                                  BCC     2$              ; IF NO CARRY, BRANCH
020723  040000  020726                  ^040000,2$
020725  115401                          INC     R1              ; INCREMENT HI WORD
020726  117402                  2$:     DEC     R2              ; DECREMENT COUNT
020727                                  BNE     1$              ; IF NO CARRY, BRANCH
020727  050000  020721                  ^050000,1$
020731  104302  002464                  MOV     ST+DBNCYL,R2    ; GET NUMBER OF READ ONLY GROUPS
020733  103202  177400                  BIC     #HIBYTE,R2      ; CLEAR UNUSED BITS
020735  107307  022640          3$:     SUB     SECGRP,R0       ; SUBTRACT NUMBER OF SECTORS/GROUP
020737                                  BCC     4$              ; IF NO BORROW, BRANCH
020737  040000  020742                  ^040000,4$
020741  117401                          DEC     R1              ; PROPOGATE BORROW
020742  117402                  4$:     DEC     R2              ; DECREMENT COUNT
020743                                  BNE     3$              ; IF INCOMPLETE, BRANCH
020743  050000  020735                  ^050000,3$
020745  107207  000001                  SUB     #1,R0           ; ADJUST FOR DBNS STARTING AT ZERO
020747                                  BCC     5$              ; IF NO BORROW, BRANCH
020747  040000  020752                  ^040000,5$
020751  117401                          DEC     R1              ; PROPOGATE BORROW
020752  104070  022635          5$:     MOV     R0,MAXDBN       ; SAVE LO ORDER WORD
020754  104010  022636                  MOV     R1,MAXDBN+1     ; SAVE HI ORDER WORD
020756                                  POP     R1              ; RESTORE REGISTERS
020756  104261                                                                  MOV (SP)+,R1
020757                                  RETURN                  ; RETURN TO INSCHR
020757  000000  000000                  ^00,0
```

```
                                .SBTTL  ********** OVERLAY MODULE SCHAR1 - SET UP SUBUNITS, CHECK TRACK/GROUP PARAMETERS
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
020761                          SCHAR1:
                                ;
                                ;       SCHAR1 WILL INITILIZE THE SUBUNIT PARAMETERS (TRACK/GROUP ONLY)
                                ;
                                ;
                                        .ENABL  LSB
020761                          PUSH    R4              ; SAVE SUBUNIT POINTER
020761  100464                                                          MOV R4,-(SP)
020762  104141                  MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
020763                          ASSUME  S.PARM,0
020763  102201  004000          BIT     #RONLY,R1       ; SEE IF READ ONLY
020765                          BEQ     5$              ; IF NOT, BRANCH
020765  010000  021040          ^010000,5$
020767  102201  002000          BIT     #WONLY,R1       ; SEE IF WRITE ONLY IS SET TOO
021071                          BEQ     6$              ; IF NOT, BRANCH
021071  010000  021012          ^010000,6$
020773                          DEVFTL  28              ; SETUP ERROR
020773  104200  002415  001601                                          MOV     #ER28,HRQ.04
020776  104202  047674                                                  MOV     #28!FTLDEV+4000.,R2
021000  104020  001577                                                  MOV     R2,HRQ.02
021002  104200  021002  001576                                          MOV     #.,HRQ.01
021005  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
021010                          BR      4$              ; BRANCH TO REPORT
021010  000000  021060          ^00,4$
021012  102201  040000  6$:     BIT     #INITW,R1       ; SEE IF INITIAL WRITE
021014                          BEQ     5$              ; IF NOT, BRANCH
021014  010000  021040          ^010000,5$
021016  103201  040000          BIC     #INITW,R1       ; NO INITIAL WRITE
021020  100141                  MOV     R1,(R4)         ; SAVE PARAMETERS
021021                          ASSUME  S.PARM,0
021021                          MSSG    4               ; REPORT NO INITIAL WRITE
021021  104200  005701  001577                                          MOV     #MS4,HRQ.02
021024  100467                                                                  MOV R0,-(SP)
021025  104650  000063  001576                                          MOV     U.UNUM(R5),HRQ.01
021030  105650  000050  001576                                          ADD     U.SUBU(R5),HRQ.01
021033  104207  060015                                                  MOV     #MESSAG,R0
021035  020000  001543          ^020000,HOSTRQ
021037  104267                                                                  MOV (SP)+,R0
021040                  5$:
                                ;       BIT     #DCYLS,R1       ; SEE IF IN DIAGNOSTIC AREA
                                ;       BEQ     7$              ; IF NOT, BRANCH
                                ;       MSSG    5               ; REPORT ASSUMPTION
                                ;7$:
021040  104041                  MOV     R4,R1           ; R1 NOW POINTS TO SUBUNIT DATASTRUCTURE
021041                          CALL    CHKBB           ; CHECK BAD BLOCKS
021041  020000  021155          ^020000,CHKBB
021043  115002                  TST     R2              ; ANY ERRORS ?
021044                          BNE     4$              ; IF SO, BRANCH
021044  050000  021060          ^050000,4$
```

```
     57 021046  104117                                MOV     (R1),R0         ; GET SUBUNIT CHARACTERISTICS
     58 021047                                      ASSUME  S.PARM,0
     59 021047  102207  000040                      BIT     #BEUSED,R0      ; SEE IF BEGIN/END SETS WERE USED
     60 021051                                      BNE     1$              ; IF SO, SKIP EVERYTHING
        021051  050000  021107                      ^O050000,1$
     61 021053                                      CALL    CHKTG           ; COMPUTE TRACK/GROUP DATABASE
        021053  020000  021346                      ^O020000,CHKTG
     62 021055  115002                              TST     R2              ; SEE IF ANY ERRORS
     63 021056                                      BEQ     1$              ; IF NOT, BRANCH
        021056  010000  021107                      ^O010000,1$
     64 021060  104650  000063  001600  4$:         MOV     U.UNUM(R5),HRQ.03       ; GET STARTING UNIT NUMBER
     65 021063  105650  000050  001600              ADD     U.SUBU(R5),HRQ.03       ; ADD OFFSET
     66 021066  104307  001575                      MOV     HRQ.RQ,R0       ; SET UP TO REPORT ERROR
     67 021070                                      CALL    HOSTRQ          ; REPORT
        021070  020000  001543                      ^O020000,HOSTRQ
     68 021072  101200  000002  003006              BIS     #DIE,M.PARM     ; FLAG INITIALIZATION ERROR
     69 021075  104201  000001                      MOV     #U.SUBP,R1      ; R1 WILL POINT AT SUBUNIT POINTERS
     70 021077  105051                              ADD     R5,R1           ; R1 POINTS AT SUBUNIT POINTERS
     71 021100  105651  000050                      ADD     U.SUBU(R5),R1   ; R1 POINTS AT POINTER TO SUBUNIT BEING HANDLED
     72 021102  104117                              MOV     (R1),R0         ; R0 POINTS TO SUBUNIT BEGING HANDLED
     73 021103  104203  100000                      MOV     #DROP,R3        ; MOVE DROP FLAG TO R3
     74 021105  100173                              MOV     R3,(R0)         ; DROP THIS SUBUNIT
     75 021106                                      ASSUME  S.PARM,0
     76 021106  114002                              CLR     R2              ; NO ERRORS
     77 021107  104653  000050          1$:         MOV     U.SUBU(R5),R3   ; GET SUBUNIT OFFSET
     78 021111  115403                              INC     R3              ; NEXT SUBUNIT
     79 021112  100653  000050                      MOV     R3,U.SUBU(R5)   ; SAVE
     80 021114  106203  000003                      CMP     #3,R3           ; SEE IF VALID OFFSET
     81 021116                                      BCS     23$             ; IF NOT, BRANCH
        021116  040000  021122                      ^O040000,.+3
        021120  000000  021135                      ^O0,23$
     82 021122  105203  000001                      ADD     #U.SUBP,R3      ; WILL POINT TO NEXT SUBUNIT POINTER
     83 021124  105053                              ADD     R5,R3           ; ADD POINTER TO UNIT DATABASE
     84 021125  104133                              MOV     (R3),R3         ; GET POINTER
     85 021126                                      BMI     1$              ; IF NO SUBUNIT, BRANCH
        021126  070000  021107                      ^O070000,1$
     86 021130  104131                              MOV     (R3),R1         ; GET SUBUNIT PARAMETERS
     87 021131                                      ASSUME  S.PARM,0
     88 021131                                      BMI     1$              ; IF DROPPED, BRANCH
        021131  070000  021107                      ^O070000,1$
     89 021133                                      ASSUME  DROP,100000
     90 021133                                      BR      22$             ; EXIT
        021133  000000  021140                      ^O0,22$
     91 021135  114003                  23$:        CLR     R3              ; NO SUBUNITS
     92 021136  100653  000050                      MOV     R3,U.SUBU(R5)   ; SAVE
     93 021140  115003                  22$:        TST     R3              ; ANY?
     94 021141                                      BNE     2$              ; IF SO, BRANCH
        021141  050000  021147                      ^O050000,2$
     95 021143  104207  021707                      MOV     #GETSER,R0      ; GETSER NEXT MODULE
     96 021145                                      BR      3$              ; EXIT
        021145  000000  021151                      ^O0,3$
     97 021147  104207  017514          2$:         MOV     #SCHAR0,R0      ; GO BACK TO FIRST CHARACTERISTICS MODULE
     98 021151  114001                  3$:         CLR     R1              ; IMMIDATE CALL
     99 021152                                      POP     R4              ; RESTORE
        021152  104264                                                                  MOV (SP)+,R4
    100 021153                                      BR      JMPRET          ; RETURN TO SEQUENCER
        021153  000000  003746                      ^O0,JMPRET
```

```
    101                                         .DSABL  LSB
```

```
                                .SBTTL  CHKBB  - CHECK THE BAD BLOCKS FOR ERRORS
021155                          CHKBB:
                                ;
                                ;       CHECK THE BAD BLOCKS TO ASSURE THEY DON'T EXCEED THE MAXIMUM BLOCK
                                ;       NUMBER
                                ;
021155                                  PUSH    R5              ; SAVE THE REGISTERS
021155  100465                                                                  MOV R5,-(SP)
021156  104613  000012                  MOV     S.BADP(R1),R3   ; GET POINTER TO BAD BLOCKS
021160                                  BEQ     5$              ; IF NO BAD BLOCKS, BRANCH
021160  010000  021342                  ^O10000,5$
021162  104632  000001          1$:     MOV     1(R3),R2        ; IS THIS THE LAST BAD BLOCK?
021164                                  BMI     2$              ; IF SO, BRANCH
021164  070000  021223                  ^O70000,2$
021166  104032                          MOV     R3,R2           ; R2 WILL POINT TO NEXT BAD BLOCK
021167  105202  000002                  ADD     #2,R2           ; R2 POINTS TO NEXT BAD BLOCK
021171                                  CALL    CBB2            ; COMPARE
021171  020000  022773                  ^O20000,CBB2
021173                                  BCC     7$              ; IF ERROR, BRANCH
021173  040000  021201                  ^O40000,7$
021175  105203  000002                  ADD     #2,R3           ; POINT TO NEXT BAD BLOCK
021177                                  BR      1$              ; LOOP
021177  000000  021162                  ^O0,1$
021201                          7$:     DEVFTL  53,#SER20       ; DUPLICATE BAD BLOCKS
021201  104200  003465  001601                                                  MOV     #ER53,HRQ.04
021204  104200  004047  001602                                                  MOV     #SER20,HRQ.05
021207  104202  047725                                                          MOV     #53!FTLDEV+4000.,R2
021211  104020  001577                                                          MOV     R2,HRQ.02
021213  104200  021213  001576                                                  MOV     #.,HRQ.01
021216  104200  060014  001575                                                  MOV     #ERRMC,HRQ.RQ
021221                                  BR      6$              ; BRANCH
021221  000000  021343                  ^O0,6$
021223  104117                  2$:     MOV     (R1),R0         ; GET SUBUNIT PARAMETERS
021224                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
021224  102207  020000                  BIT     #DCYLS,R0       ; SEE IF DIAGNOSTIC CYLINDERS IN USE
021226                                  BNE     3$              ; IF SO, BRANCH
021226  050000  021303                  ^O50000,3$
021230  104612  000007                  MOV     S.SCHR(R1),R2   ; R2 POINTS TO SUBUNIT CHARACTERISTICS
021232  105202  000012                  ADD     #LBNHST,R2      ; R2 POINTS TO NUMBER OF LBNS IN HOST AREA
021234                                  CALL    CBB2            ; COMPARE
021234  020000  022773                  ^O20000,CBB2
021236                                  BCC     8$              ; IF ERROR, BRANCH
021236  040000  021242                  ^O40000,8$
021240                                  BR      5$              ; EXIT WITH NO ERROR
021240  000000  021342                  ^O0,5$
021242  104125                  8$:     MOV     (R2),R5         ; GET LO ORDER MAX LBN
021243  107205  000001                  SUB     #1,R5           ; DECREMENT COUNT
021245  100125                          MOV     R5,(R2)         ; SAVE
021246                                  BCC     9$              ; IF NO CARRY, BRANCH
021246  040000  021255                  ^O40000,9$
021250  104625  000001                  MOV     1(R2),R5        ; GET HI ORDER
021252  117405                          DEC     R5              ; PROPOGATE CARRY
021253  100625  000001                  MOV     R5,1(R2)        ; SAVE
021255                          9$:     DEVFTL  54,<#SER20,(R2)+,(R2)>  ; BAD BLOCK > MAXIMUM
021255  104200  003502  001601                                                  MOV     #ER54,HRQ.04
021260  104200  004047  001602                                                  MOV     #SER20,HRQ.05
021263  104220  001603                                                          MOV     (R2)+,HRQ.06
```

```
                                                                        MOV     (R2),HRQ.07
                                                                        MOV     #54!FTLDEV+4000.,R2
                                                                        MOV     R2.HRQ.02
                                                                        MOV     #.,HRQ.01
                                                                        MOV     #ERRMC,HRQ.RQ
 37 021301                                      BR      6$              ; ERROR EXIT
    021301  000000  021343                     ^O0,6$
 38 021303  104032                      3$:     MOV     R3,R2           ; R2 POINTS TO LAS BAD BLOCK
 39 021304  104203  022635                     MOV     #MAXDBN,R3      ; R3 POINTS TO MAXIMUM DBN
 40 021306                                      CALL    CBB2            ; COMPARE
    021306  020000  022773                     ^O20000,CBB2
 41 021310                                      BCC     5$              ; IF NO ERROR, BRANCH
    021310  040000  021342                     ^O40000,5$
 42 021312                              4$:     DEVFTL  54,<#SER20,MAXDBN,MAXDBN+1> ; BAD BLOCK > MAXIMUM
    021312  104200  003502  001601                                      MOV     #ER54,HRQ.04
    021315  104200  004047  001602                                      MOV     #SER20,HRQ.05
    021320  104300  022635  001603                                      MOV     MAXDBN,HRQ.06
    021323  104300  022636  001604                                      MOV     MAXDBN+1,HRQ.07
    021326  104202  047726                                             MOV     #54!FTLDEV+4000.,R2
    021330  104020  001577                                             MOV     R2,HRQ.02
    021332  104200  021332  001576                                     MOV     #.,HRQ.01
    021335  104200  060014  001575                                     MOV     #ERRMC,HRQ.RQ
 43 021340                                      BR      6$              ; BRANCH TO EXIT
    021340  000000  021343                     ^O0,6$
 44 021342  114002                      5$:     CLR     R2              ; FLAG AS NO ERRORS
 45 021343                              6$:     POP     R5              ; RESTORE THE REGISTERES
    021343  104265                                                                      MOV (SP)+,R5
 46 021344                                      RETURN                  ; RETURN TO INSCHR
    021344  000000  000000                     ^O0,0
```

```
                                .SBTTL  CHKTG  - CHECK THE TRACK/GROUPS FOR ERRORS, AND CONVERT TO BN'S
021346                          CHKTG:
                                ;
                                ;       CONVERT THE TRACKS/GROUPS TO SECTOR COUNTS, AND FIND THE MAXIMUM
                                ;       NUMBER OF 'LOOPS' THAT SETUP CAN RUN THROUGH TO TEST ALL
                                ;       TRACKS ON THE TESTED CYLINDERS ONLY
                                ;
                                ;
                                ;       FIRST COPY THE TRACKS/GROUPS TO A TEMP LOCATION
                                ;
021346                                  PUSH    R5                      ; SAVE R5
021346  100465                                                                  MOV R5,-(SP)
021347  104207  022642                  MOV     #TGS,R0                 ; POINT TO TEMPORARY STORAGE LOCATION
021351  104202  000017                  MOV     #S.TGSS,R2              ; R2 WILL POINT TO T/G LIST
021353  105012                          ADD     R1,R2                   ; R2 POINTS TO T/G LIST
021354  104225                  1$:     MOV     (R2)+,R5                ; GET WORD
021355  100275                          MOV     R5,(R0)+                ; SAVE IN TEMP LOCATION
021356                                  BPL     1$                      ; COPY ENTIRE LIST
021356  030000  021354                  ^030000,1$
                                ;
                                ;       NOW MOVE THE STARTING CYLINDER TO THE INITIAL OFFSET AREA
                                ;
021360  104612  000016                  MOV     S.TGOF+1(R1),R2         ; GET HI ORDER STARTING CYL
021362  103202  170000                  BIC     #^CHBHINB,R2            ; CLEAR UNUSED BITS
021364  100612  000016                  MOV     R2,S.TGOF+1(R1)         ; SAVE
                                ;
                                ;       CHECK THAT THE MAXIMUM TRACK/GROUP IS VALID
                                ;
021366  104117                          MOV     (R1),R0                 ; GET SUBUNIT PARAMETERS
021367                                  ASSUME  S.PARM,0                ; ASSUME THAT S.PARM IS ZERO
021367  104642  000007                  MOV     S.SCHR(R4),R2           ; R2 POINTS TO SUBUNIT CHAR
021371  102207  000020                  BIT     #TRACKS,R0              ; SEE IF PROCESSING GROUPS OR TRACKS
021373                                  BNE     2$                      ; IF TRACKS, BRANCH
021373  050000  021404                  ^050000,2$
021375  104627  000002                  MOV     GRPCYL(R2),R0           ; GET GROUPS PER CYLINDER
021377  104200  005513  021663          MOV     #GRP$,TG$               ; PRINT 'GROUP' IF ERROR
021402                                  BR      3$                      ; BRANCH
021402  000000  021411                  ^00,3$
021404  104627  000003          2$:     MOV     TRKGRP(R2),R0           ; GET TRACKS PER GROUP
021406  104200  005507  021663          MOV     #TRK$,TG$               ; PRINT 'TRACK' IF ERROR
021411  103207  177400          3$:     BIC     #HIBYTE,R0              ; CLEAR UNUSED BITS
021413  104202  022642                  MOV     #TGS,R2                 ; POINT TO TRACK/GROUP LIST
021415  104223                  4$:     MOV     (R2)+,R3                ; SEE IF THIS WORD IS END-OF-LIST
021416                                  BPL     4$                      ; IF NOT, BRANCH
021416  030000  021415                  ^030000,4$
021420  103203  177400                  BIC     #HIBYTE,R3              ; CLEAR EOL FLAG
021422  106037                          CMP     R3,R0                   ; SEE IF WITHIN LIMITS
021423                                  BMI     5$                      ; IF SO, BRANCH
021423  070000  021455                  ^070000,5$
021425  117407                          DEC     R0                      ; ADJUST T/G MAX FOR 0-N
021426                                  DEVFTL  58,<#SER20,TG$,R0>      ; REPORT ERROR
021426  104200  003717  001601                  MOV     #ER58,HRQ.04
021431  104200  004047  001602                  MOV     #SER20,HRQ.05
021434  104300  021663  001603                  MOV     TG$,HRQ.06
021437  104070  001604                          MOV     R0,HRQ.07
021441  104202  047732                          MOV     #58!FTLDEV+4000.,R2
021443  104020  001577                          MOV     R2,HRQ.02
```

CHKTG - CHECK THE TRACK/GROUPS FOR ERRORS, AND CONVERT TO BN'

```
        021445  104200  021445  001576                          MOV     #.,HRQ.01
        021450  104200  060014  001575                          MOV     #ERRMC,HRQ.RQ
     46 021453                          BR      17$             ; EXIT
        021453  000000  021660          ^O0,17$
```

```
                                        ;
                                        ;       NOW COMPUTE THE NUMBER OF SECTORS FROM THE START OF THE CYLINDER/GROUP
                                        ;       TO THE FIRST TRACK/GROUP TO TEST
                                        ;
021455  104207  022642          5$:   MOV     #TGS,R0                 ; R0 POINTS TO TRACK/GROUP LIST
021457  104203  000015                MOV     #S.TGOF,R3              ; R3 WILL POINT TO T/G INITIAL OFFSET
021461  105013                        ADD     R1,R3                   ; R3 POINTS TO T/G INITIAL OFFSET
021462  104172                        MOV     (R0),R2                 ; GET FIRST T/G
021463  103202  177400                BIC     #HIBYTE,R2              ; CLEAR UNUSED BITS
021465                                CALL    COMPDP                  ; COMPUTE SECTORS FROM CYL TO INIT T/G
021465  020000  021664                ^O020000,COMPDP
021467  105135                        ADD     (R3),R5                 ; ADD TO INITIAL OFFSET
021470  100235                        MOV     R5,(R3)+                ; MOVE BACK
021471                                BCC     6$                      ; IF NO CARRY, BRANCH
021471  040000  021476                ^O040000,6$
021473  104135                        MOV     (R3),R5                 ; GET HI ORDER WORD
021474  115405                        INC     R5                      ; PROPOGATE CARRY
021475  100135                        MOV     R5,(R3)                 ; SAVE
021476  115403                  6$:   INC     R3                      ; POINT TO NEXT AREA
                                        ;
                                        ;       NOW COMPUE HOW MANY SECTORS FROM THE LAST TRACK/GROUP TO TEST TO THE
                                        ;       NEXT TRACK/GROUP TO TEST
                                        ;
021477  104174                  7$:   MOV     (R0),R4                 ; GET TRACK/GROUP
021500                                BMI     8$                      ; IF EOL, BRANCH
021500  070000  021545                ^O070000,8$
021502  104672  000001                MOV     1(R0),R2                ; GET NEXT TRACK
021504  103202  177400                BIC     #HIBYTE,R2              ; CLEAR EOL FLAG, IF ANY
021506  107272                        SUB     (R0)+,R2                ; GET HOW MANY T/G'S BETWEEN LAST/NEXT
021507                                BEQ     20$                     ; IF ZERO, ERROR
021507  010000  021513                ^O010000,20$
021511                                BPL     18$                     ; SHOULD BE AT LEAST 1 T/G BETWEEN
021511  030000  021540                ^O030000,18$
021513                          20$:  DEVFTL  59,<#SER20,TG$>         ; REPORT ERROR
021513  104200  003750  001601                      MOV     #ER59,HRQ.04
021516  104200  004047  001602                      MOV     #SER20,HRQ.05
021521  104300  021663  001603                      MOV     TG$,HRQ.06
021524  104202  047733                            MOV     #59!FTLDEV+4000.,R2
021526  104020  001577                            MOV     R2,HRQ.02
021530  104200  021530  001576                    MOV     #.,HRQ.01
021533  104200  060014  001575                    MOV     #ERRMC,HRQ.RQ
021536                                BR      17$                     ; EXIT
021536  000000  021660                ^O00,17$
021540                          18$:  CALL    COMPDP                  ; COMPUTE HOW MANY SECTORS TO NEXT T/G
021540  020000  021664                ^O020000,COMPDP
021542  100235                        MOV     R5,(R3)+                ; STORE THE OFFSET
021543                                BR      7$                      ; LOOP
021543  000000  021477                ^O00,7$
                                        ;
                                        ;       NOW COMPUTE HOW MANY SECTORS FROM THE LAST T/G TO TEST TO THE FIRST
                                        ;
021545  104115                  8$:   MOV     (R1),R5                 ; GET SUBUNIT PARAMETERS
021546                                ASSUME  S.PARM,0                ; ASSUME THAT S.PARM IS ZERO
021546  104612  000007                MOV     S.SCHR(R1),R2           ; R2 POINTS TO SUBUNIT CHAR
021550  102205  000020                BIT     #TRACKS,R5              ; SEE IF USING GROUPS OR TRACKS
021552                                BNE     9$                      ; IF TRACKS, BRANCH
021552  050000  021560                ^O050000,9$
```

```
CHKTG  - CHECK THE TRACK/GROUPS FOR ERRORS, AND CONVERT TO BN'

     021554  104622  000002                  MOV     GRPCYL(R2),R2           ; GET GROUPS/CYL
     021556                                  BR      10$                     ; BRANCH
     021556  000000  021562                  ^O00,10$
     021560  104622  000003          9$:     MOV     TRKGRP(R2),R2           ; GET TRACKS/GROUP
     021562  105302  022642          10$:    ADD     TGS,R2                  ; ADD STARTING T/G
     021564  103202  177400                  BIC     #HIBYTE,R2              ; CLEAR UNUSED BITS
     021566  103204  177400                  BIC     #HIBYTE,R4              ; CLEAR UNUSED BITS
     021570  107042                          SUB     R4,R2                   ; FIND HOW MANY T/G'S TO SKIP
     021571                                  CALL    COMPDP                  ; COMPUTE HOW MANY SECTORS
     021571  020000  021664                  ^O20000,COMPDP
     021573  100235                          MOV     R5,(R3)+                ; STORE IN LIST
     021574  114005                          CLR     R5                      ; SET UP FOR MARKING EOL
     021575  100235                          MOV     R5,(R3)+                ; STORE EOL
                                     ;
                                     ;      NOW SEE HOW MANY TIMES THE SETUP ROUTINE CAN GO THROUGH THE LIST
                                     ;      WITHOUT EXCEEDING THE MAXIMUM SECTOR NUMBER
                                     ;
     021576  104207  000013                  MOV     #S.BESS,R0              ; R0 WILL POINT TO THE MAX SEC NUMBER
     021600  105017                          ADD     R1,R0                   ; R0 POINTS TO THE MAX SECTOR NUMBER
     021601  104674  000002                  MOV     S.TGOF-S.BESS(R0),R4    ; R4 HAS LO ORDER INITIAL OFFSET
     021603  104675  000003                  MOV     S.TGOF-S.BESS+1(R0),R5  ; R5 HAS HI ORDER INITIAL OFFSET
     021605                                  PUSH    R1                      ; SAVE R1
     021605  100461                                                                  MOV R1,-(SP)
     021606  114002                          CLR     R2                      ; CLEAR LO ORDER MAX COUNT
     021607  114003                          CLR     R3                      ; CLEAR HI ORDER MAX COUNT
     021610  104201  000004                  MOV     #S.TGSS-S.BESS,R1       ; R1 WILL POINT TO SECTOR OFFSET AREA
     021612  105071                          ADD     R0,R1                   ; R1 POINTS TO SECTOR OFFSET AREA
     021613  100464                  13$:    MOV     R4,-(SP)                ; SAVE LO ORDER TOTAL ON STACK
     021614  104214                          MOV     (R1)+,R4                ; GET LO ORDER SECTOR OFFSET
     021615                                  BNE     14$                     ; IF NOT EOL, BRANCH
     021615  050000  021623                  ^O50000,14$
     021617  104071                          MOV     R0,R1                   ; R1 WILL POINT TO START OF OFFSET LIST
     021620  105201  000004                  ADD     #S.TGSS-S.BESS,R1       ; R1 POINTS TO START OF OFFSET LIST
     021622  104214                          MOV     (R1)+,R4                ; GET NEW OFFSET
     021623  105264                  14$:    ADD     (SP)+,R4                ; ADD TO LOW ORDER SECTOR TOTAL
     021624                                  BCC     15$                     ; IF NO CARRY, BRANCH
     021624  040000  021627                  ^O40000,15$
     021626  115405                          INC     R5                      ; PROPOGATE CARRY
     021627  106675  000001          15$:    CMP     1(R0),R5                ; SEE IF MAX EXCEEDED
     021631                                  BCS     16$                     ; IF SO, BRANCH
     021631  040000  021635                  ^O40000,..+3
     021633  000000  021653                  ^O00,16$
     021635                                  BNE     11$                     ; IF YOUR SURE IT'S OK, BRANCH
     021635  050000  021644                  ^O50000,11$
     021637  106174                          CMP     (R0),R4                 ; SEE IF MAX EXCEEDED
     021640                                  BCS     16$                     ; IF SO, BRANCH
     021640  040000  021644                  ^O40000,..+3
     021642  000000  021653                  ^O00,16$
     021644  105202  000001          11$:    ADD     #1,R2                   ; ADD 1 TO COUNT
     021646                                  BCC     13$                     ; IF NO CARRY, BRANCH
     021646  040000  021613                  ^O40000,13$
     021650  115403                          INC     R3                      ; PROPOGATE CARRY
     021651                                  BR      13$                     ; BRANCH
     021651  000000  021613                  ^O00,13$
     021653                          16$:    POP     R1                      ; RESTORE R1
     021653  104261                                                                  MOV (SP)+,R1
     021654  100172                          MOV     R2,(R0)                 ; STORE LOW ORDER MAXIMUM COUNT
```

```
     86 021655  100673  000001                    MOV     R3,1(R0)        ; STORE HI MAXIMUM COUNT
     87 021657  114002                            CLR     R2              ; CLEAR R2
     88 021660                          17$:      POP     R5              ; RESTORE
        021660  104265                                                              MOV (SP)+,R5
     89 021661                                    RETURN
        021661  000000  000000                    ^O0,0
     90
     91 021663                          TG$:      .BLKW   1
```

COMPDP - CALCULATE SECTORS/TRACKS OR SECTORS/GROUPS

```
                                        .SBTTL  COMPDP - CALCULATE SECTORS/TRACKS OR SECTORS/GROUPS
021664                                  COMPDP:
                                        ;
                                        ;       COMPUTE HOW MANY SECTORS ARE IN THE NUMBER OF TRACKS OR GROUPS
                                        ;       PASSED IN R2
                                        ;
021664  114005                                  CLR     R5                      ; CLEAR RUNNING TOTAL
021665  104114                                  MOV     (R1),R4                 ; GET SUBUNIT PARAMETERS
021666                                          ASSUME  S.PARM,0                ; ASSUME THAT S.PARM IS ZERO
021666                                          BR      3$                      ; SEE IF IMMIDATE EXIT
021666  000000  021702                          ^O00,3$
021670  102204  000020                  1$:     BIT     #TRACKS,R4              ; SEE IF USING GROUPS OR TRACKS
021672                                          BNE     2$                      ; IF TRACKS, BRANCH
021672  050000  021700                          ^O50000,2$
021674  105305  022640                          ADD     SECGRP,R5               ; ADD SECTORS/GROUP TO RUNNING TOTAL
021676                                          BR      3$                      ; BRANCH
021676  000000  021702                          ^O00,3$
021700  105305  022637                  2$:     ADD     SECTRK,R5               ; ADD SECTORS/TRACK TO RUNNING TOTAL
021702  11740?                          3$:     DEC     R2                      ; DECREMENT COUNT
021703                                          BPL     1$                      ; IF INCOMPLETE, BRANCH
021703  030000  021670                          ^O30000,1$
021705                                          RETURN                          ; RETURN TO CALLING PROGRAM
021705  000000  000000                          ^O00,0
```

```
                                        .SBTTL  ********** OVERLAY MODULE GETSER - GET THE HDA AND DRIVE SERIAL NUMBER
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;************ *****************************************************************
                                        ;
                                        ;
                                        ;
021707                                  GETSER:
                                        ;
                                        ;       GET HDA AND DRIVE SERIAL NUMBER (IF POSSIBLE) AND RETURN TO HOST
                                        ;
                                                .ENABL  LSB
021707                                          PUSH    R4                      ; SAVE SUBUNIT POINTER
021707  100464                                                                          MOV R4,-(SP)
021710  104650  000063  022416                  MOV     U.UNUM(R5),SUNIT        ; SAVE UNIT NUMBER
021713  104651  000063                          MOV     U.UNUM(R5),R1           ; R1 HAS UNIT NUMBER
021715  105201  000003                          ADD     #3,R1                   ; R1 HAS LAST UNIT NUMBER
021717  104207  000001                          MOV     #U.SUBP,R0              ; R0 WILL POINT TO SUBUNIT POINTERS
021721  105057                                  ADD     R5,R0                   ; R0 POINTS TO SUBUNIT POINTERS
021722  104274                          11$:    MOV     (R0)+,R4                ; R4 POINTS TO SUBUNIT
021723                                          BMI     12$                     ; IF NO SUBUNIT, BRANCH
021723  070000  022377                          ^O70000,12$
021725  104203  001750                          MOV     #1000.,R3               ; INITILIZE SEEK COUNTER
021727  100643  000001                          MOV     R3,S.SEEK(R4)           ; SAVE SEEK COUNTER
021731                                          PUSH    <R0,R1>                 ; SAVE POINTER AND COUNT
021731  100467                                                                          MOV R0,-(SP)
021732  100461                                                                          MOV R1,-(SP)
                                        .SBTTL  REPSER - FIND AND REPORT DRIVE AND HDA SERIAL NUMBER
                                        ;REPSER
                                        ;
                                        ;       FIND AND REPORT DRIVE AND HDA SERIAL NUMBER.  IF ERROR OCCURRS,
                                        ;       REPORT SERIAL NUMBER AS ZERO.  NO ERROR IS RETURNED FROM THIS ROUTINE
                                        ;
021733                                          CALL    BULDUM                  ; BUILD DUMMY SDI BLOCK
021733  020000  002552                          ^O020000,BULDUM
021735  104641  000007                          MOV     S.SCHR(R4),R1           ; R1 POINTS TO SUBUNIT CHARACTERISTICS
021737  104610  000000  002756                  MOV     LBNCYL(R1),STACYL       ; MOVE LO STARTING CYL
021742  104610  000001  002757                  MOV     LBNCYL+1(R1),STACYL+1   ; MOVE HI STARTING CYL
021745  104617  000004                          MOV     RBNTRK(R1),R0           ; GET RBNS/TRACK
021747  103207  177600                          BIC     #HIBYTE!200,R0          ; CLEAR UNUSED BITS
021751  105617  000011                          ADD     LBNTRK(R1),R0           ; ADD LBNS/TRACK
021753  103207  177400                          BIC     #HIBYTE,R0              ; CLEAR UNUSED BITS
021755  104070  002762                          MOV     R0,LRDTRK               ; MOVE TO SECTORS PER TRACK
021757  114000  002763                          CLR     RBNFLG                  ; NO RBN FLAG
021761  104657  000057                          MOV     U.COPY(R5),R0           ; GET NUMBER OF XBN COPIES
021763  105077                                  ADD     R0,R0                   ; DOUBLE
021764  104070  022417                          MOV     R0,SERRTY               ; RETRY 2 TIMES THE NUMBER OF COPIES
021766  114007                          1$:     CLR     R0                      ; TO SET UP COPIES AND SETUP CALC
021767  100657  000060                          MOV     R0,U.CCOP(R5)           ; SAVE
021771  104641  000007                          MOV     S.SCHR(R4),R1           ; GET POINTER TO SUBUNIT CHARACTERISTICS
021773  114000  002760                          CLR     CURBN                   ; PUT LO ORDER XBN IN CURBN
021775  114000  002761                          CLR     CURBN+1                 ; PUT HI ORDER XBN IN CURBN+1
022001  104641  000007                  2$:     MOV     #CALCSC,R0              ; POINT TO CALCULATION AREA
022003  060020                                  MOV     S.SCHR(R4),R1           ; POINT TO SUBUNIT CHARACTERISTICS
022004                                          XFC     CVT                     ; CONVERT
                                                CALL    RECOVR                  ; CLEAR ALL POSSIBLE ERRORS
```

```
     022004  020000  004505                          ^020000,RECOVR
  71 022006  115002                                  TST     R2              ; SEE IF ERROR OCCURRED
  72 022007                                          BNE     8$              ; IF SO, BRANCH
     022007  050000  022257                          ^050000,8$
  73 022011  104203  002415                          MOV     #CR.INR,R3      ; SET UP FOR RECALIBRATE
  74 022013                                          CALL    TALK            ; RECALIBRATE DRIVE
     022013  020000  001670                          ^020000,TALK
  75 022015  115002                                  TST     R2              ; SEE IF ERROR OCCURRED
  76 022016                                          BNE     8$              ; IF SO, BRANCH
     022016  050000  022257                          ^050000,8$
  77 022020  104300  002764  002433                  MOV     CYL,LOCYL       ; MOVE LO CYL TO SEEK COMMAND
  78 022023  104300  002765  002434                  MOV     CYL+1,LOCYL+1   ; MOVE TO SEEK COMMAND
  79 022026  104300  002766  002435                  MOV     GROUP,LOCYL+2   ; MOVE GROUP TO SEEK COMMAND
  80 022031  104203  002376                          MOV     #CR.SEK,R3      ; POINT TO SEEK COMMAND
  81 022033                                          CALL    TALK            ; SEND SEEK
     022033  020000  001670                          ^020000,TALK
  82 022035  115002                                  TST     R2              ; SEE IF ERROR OCCURRED
  83 022036                                          BNE     8$              ; IF SO, BRANCH
     022036  050000  022257                          ^050000,8$
  84 022040  104651  000011                          MOV     U.MSTO(R5),R1   ; GET MASTER SEEK TIMEOUT
  85 022042  100651  000005                          MOV     R1,U.TIMH(R5)   ; SAVE
  86 022044  114001                                  CLR     R1              ; FOR LOW ORDER TIMEOUT
  87 022045  100651  000006                          MOV     R1,U.TIML(R5)
  88 022047  104201  000310              3$:         MOV     #200.,R1        ; INNER LOOP TIMEOUT
  89 022051  117401                      4$:         DEC     R1              ; DECREMENT INNER COUNTER
  90 022052                                          BNE     4$              ; IF UNEXPIRED, BRANCH
     022052  050000  022051                          ^050000,4$
  91 022054                                          CALL    RTDSL           ; GET REAL TIME DRIVE STATE
     022054  020000  001453                          ^020000,RTDSL
  92 022056  115002                                  TST     R2              ; SEE IF ERROR OCCURRED
  93 022057                                          BNE     8$              ; IF SO, BRANCH
     022057  050000  022257                          ^050000,8$
  94 022061  115001                                  TST     R1              ; SEE IF READ/WRITE READY IS ASSERTED
  95 022062                                          BMI     5$              ; IF SO, BRANCH
     022062  070000  022123                          ^070000,5$
  96 022064                                          ASSUME  RWRDY,100000
  97 022064  114000  001577                          CLR     HRQ.02          ; SETUP TO TALK TO HOST
  98 022066  114000  001600                          CLR     HRQ.03          ; SETUP TO TALK TO HOST
  99 022070  114000  001601                          CLR     HRQ.04          ; SETUP TO TALK TO HOST
 100 022072  104300  022416  001576                  MOV     SUNIT,HRQ.01    ; MOVE UNIT NUMBER INTO SEND BUFFER
 101 022075  104207  060007                          MOV     #T4SOFT,R0      ; MOVE REQUEST NUMBER TO R0
 102 022077                                          CALL    HOSTRQ          ; TELL THE HOST I'M ALIVE
     022077  020000  001543                          ^020000,HOSTRQ
 103 022101  104651  000006                          MOV     U.TIML(R5),R1   ; GET TIMEOUT
 104 022103  107201  000001                          SUB     #1,R1           ; DECREMENT TIMEOUT
 105 022105  100651  000006                          MOV     R1,U.TIML(R5)   ; SAVE
 106 022107                                          BCC     3$              ; LOOP IF INCOMPLETE
     022107  040000  022047                          ^040000,3$
 107 022111  104651  000005                          MOV     U.TIMH(R5),R1   ; GET HI ORDER TIMEOUT
 108 022113  107201  000001                          SUB     #1,R1           ; DECREMENT TIMEOUT
 109 022115  100651  000005                          MOV     R1,U.TIMH(R5)   ; SAVE
 110 022117                                          BCC     3$              ; IF TIMOUT UNEXPIRED, BRANCH
     022117  040000  022047                          ^040000,3$
 111 022121                                          BR      8$              ; ERROR
     022121  000000  022257                          ^00,8$
 120 022123  104200  100000  022420  5$:             MOV     #RSTOP,SERCHN+RW.CPT    ; MOVE LAST CHAIN FLAG TO CHAIN
 124 022126  104300  002760  022423                  MOV     CURBN,SERCHN+RW.LOW     ; MOVE TO OUTPUT
```

```
125 022131  104641  000007                  MOV     S.SCHR(R4),R1           ; R1 POINTS TO SUBUNIT CHARACTERISTICS
126 022133  104611  000002                  MOV     HIXBN(R1),R1            ; GET HI XBN BITS
127 022135  110601                          ROR     R1                      ; ROTATE TO CORRECT POSITION
128 022136  110601                          ROR     R1                      ; ROTATE TO CORRECT POSITION
129 022137  110601                          ROR     R1                      ; ROTATE TO CORRECT POSITION
130 022140  110601                          ROR     R1                      ; ROTATE TO CORRECT POSITION
131 022141  103201  170377                  BIC     #HBLONB,R1              ; CLEAR UNUSED BITS
132 022143  101201  120000                  BIS     #HD.XBN,R1              ; MAKE A XBN
133 022145  105301  002761                  ADD     CURBN+1,R1              ; SET IN HI XBN BITS
134 022147  104010  022424                  MOV     R1,SERCHN+RW.HI         ; SAVE
135 022151  104300  002767  022425          MOV     TRACK,SERCHN+RW.CMD     ; MOVE TRACK TO CHAIN
136 022154  101200  013400  022425          BIS     #RREAL,SERCHN+RW.CMD    ; SET IN REAL TIME COMMAND
137 022157  104200  002772  022426          MOV     #DUMSDI,SERCHN+RW.SDI   ; POINT TO DUMMY SDI BLOCK
138 022162  104652  000025                  MOV     U.MASK(R5),R2           ; R2 HAS UDA PORT MASK
139 022164  104207  022420                  MOV     #SERCHN,R0              ; R0 POINTS TO CHAIN
140 022166  060012                          XFC     WAITSI                  ; WAIT FOR SECTOR OR INDEX
141 022167  115001                          TST     R1                      ; SEE IF ERROR OCCURRED
142 022170                                  BNE     8$                      ; IF SO, BRANCH
    022170  050000  022257                  ^050000,8$
146 022172  104202  000400                  MOV     #SEC512,R2              ; READ A 512 BYTE SECTOR
150 022174  060002                          XFC     XREAD                   ; READ THE SECTOR
151 022175  106201  000004                  CMP     #4,R1                   ; SEE IF XBN HEADER COMPARE FAILURE
152 022177                                  BEQ     7$                      ; IF SO, TRY NEXT COPY
    022177  010000  022233                  ^010000,7$
153 022201  115001                          TST     R1                      ; SEE IF AN ERROR OCCURRED
154 022202                                  BNE     8$                      ; IF SO, BRANCH
    022202  050000  022257                  ^050000,8$
163 022204  102200  010000  022420          BIT     #ECCFLG,SERCHN+RW.CPT   ; SEE IF BUFFER HAS AN ECC ERROR
167 022207                                  BEQ     6$                      ; IF NOT, BRANCH
    022207  010000  022223                  ^010000,6$
168 022211  104207  022420                  MOV     #SERCHN,R0              ; POINT TO CHAIN
169 022213  060015                          XFC     ECC                     ; CORRECT THE BUFFER
170 022214  115001                          TST     R1                      ; SEE IF ERROR
171 022215                                  BNE     7$                      ; IF SO, BRANCH
    022215  050000  022233                  ^050000,7$
172 022217  106657  000032                  CMP     U.ECCT(R5),R0           ; SEE IF CORRECTIONS EXCEED THRESHOLD
173                             ;
174                             ;           NOTE: I ALLOW 1 ERROR MORE THAN SPEC'D ECC THRESHOLD
175                             ;
176 022221                                  BMI     7$                      ; IF SO, BRANCH
    022221  070000  022233                  ^070000,7$
177 022223  104207  023057          6$:     MOV     #SERSEC,R0              ; R0 POINTS TO BUFFER
178 022225                                  CALL    CMPEDC                  ; COMPUTE EDC OVER BUFFER
    022225  020000  001640                  ^020000,CMPEDC
179 022227  106020  023457                  CMP     R2,SERSEC+BF.EDC        ; SEE IF EDC'S MATCH
180 022231                                  BEQ     9$                      ; IF SO, YOU'VE GO IT!!!!!!!!!!!!
    022231  010000  022275                  ^010000,9$
181 022233  104651  000060          7$:     MOV     U.CCOP(R5),R1           ; GET CURRENT COPY NUMBER
182 022235  115401                          INC     R1                      ; INCREMENT COUNT
183 022236  100651  000060                  MOV     R1,U.CCOP(R5)           ; SAVE
184 022240  106651  000057                  CMP     U.COPY(R5),R1           ; SEE IF ALL COPIES TRIED
185 022242                                  BEQ     8$                      ; IF SO, BRANCH
    022242  010000  022257                  ^010000,8$
186 022244  104647  000007                  MOV     S.SCHR(R4),R0           ; R1 POINTS TO SUBUNIT CHARACTERISTICS
187 022246  105670  000010  002760          ADD     FCTSIZ(R0),CURBN        ; ADD TO CURRENT BN
188 022251                                  BCC     2$                      ; CALCULATE AND TRY NEXT SECTOR
    022251  040000  021777                  ^040000,2$
```

```
                                                ;
                                                ;       COULDN'T GET THE SERIAL NUMBER -- REPORT 0'S AND 512 DRIVE
                                                ;
022263  114000  023061                          CLR     SERSEC+2                ; ZERO HDA SER #
022265  114000  023062                          CLR     SERSEC+3                ; ZERO HDA SER #
022267  114000  023063                          CLR     SERSEC+4                ; ZERO HDA SER #
022271  114000  023064                          CLR     SERSEC+5                ; ZERO HDA SER #
022273                                          BR      10$                     ; ASSUME 512 DRIVE
        022273  000000  022341                  ^O00,10$
022275  106200  126736  023057  9$:     CMP     #MOD512,SERSEC          ; SEE IF 512 BYTE MODE DRIVE
022300                                          BEQ     10$                     ; IF SO, BRANCH
        022300  010000  022341                  ^O10000,10$
022302  104147                                  MOV     (R4),R0                 ; GET UNIT PARAMETERS
022303                                          ASSUME  S.PARM,0
022303  102207  020000                          BIT     #DCYLS,R0               ; SEE IF USING THE DIAGNOSTIC AREA
022305                                          BNE     10$                     ; IF SO, OK TO TEST, BRANCH
        022305  050000  022341                  ^O50000,10$
022307                                          DEVFTL  21,SERSEC               ; SET UP REPORT
        022307  104200  002106  001601                                                  MOV     #ER21,HRQ.04
        022312  104300  023057  001602                                                  MOV     SERSEC,HRQ.05
        022315  104202  047665                                                          MOV     #21!FTLDEV+4000.,R2
        022317  104020  001577                                                          MOV     R2,HRQ.02
        022321  104200  022321  001576                                                  MOV     #.,HRQ.01
        022324  104200  060014  001575                                                  MOV     #ERRMC,HRQ.RQ
022327  104300  022416  001600                  MOV     SUNIT,HRQ.03            ; MOVE UNIT NUMBER TO MESSAGE
022332  104307  001575                          MOV     HRQ.RQ,R0               ; GET REQUEST NUMBER
022334                                          CALL    HOSTRQ                  ; REPORT TO HOST
        022334  020000  001543                  ^O20000,HOSTRQ
022336  101200  000002  003006                  BIS     #DIE,M.PARM             ; DO NOT TEST
022341  104300  022416  001576  10$:     MOV     SUNIT,HRQ.01            ; MOVE IN UNIT NUMBER
022344  104300  022651  001577                  MOV     DSERNM,HRQ.02           ; MOVE DRIVE SERIAL NUMBER TO BUFFER
022347  104300  022652  001600                  MOV     DSERNM+1,HRQ.03         ; MOVE DRIVE SERIAL NUMBER TO BUFFER
022352  104300  022653  001601                  MOV     DSERNM+2,HRQ.04         ; MOVE DRIVE SERIAL NUMBER TO BUFFER
022355  104300  023061  001602                  MOV     SERSEC+2,HRQ.05         ; MOVE HDA SERIAL NUMBER TO BUFFER
022360  104300  023062  001603                  MOV     SERSEC+3,HRQ.06         ; MOVE HDA SERIAL NUMBER TO BUFFER
022363  104300  023063  001604                  MOV     SERSEC+4,HRQ.07         ; MOVE HDA SERIAL NUMBER TO BUFFER
022366  104300  023064  001605                  MOV     SERSEC+5,HRQ.08         ; MOVE HDA SERIAL NUMBER TO BUFFER
022371  104207  060004                          MOV     #T4UPRM,R0              ; MOVE IN REQUEST
022373                                          CALL    HOSTRQ                  ; REPORT
        022373  020000  001543                  ^O20000,HOSTRQ
022375                                          POP     <R1,R0>                 ; RESTORE POINTER AND COUNT
        022375  104261                                                                  MOV (SP)+,R1
        022376  104267                                                                  MOV (SP)+,R0
022377  115400  022416          12$:    INC     SUNIT                   ; INCREMENT ACTIVE SUBUNIT
022401  106010  022416                  CMP     R1,SUNIT                ; SEE IF ALL SUBUNITS REPORTED
022403                                          BCC     11$                     ; IF NOT, BRANCH
        022403  040000  021722                  ^O40000,11$
022405                                          POP     R4                      ; RESTORE SUBUNIT POINTER
        022405  104264                                                                  MOV (SP)+,R4
022406  114002                                  CLR     R2                      ; NO ERRORS
022407  114001                                  CLR     R1                      ; IMMIDATE CALL
```

```
    229 022410  104207  022427                  MOV     #INITD,R0           ; INITD NEXT ROUTINE
    230 022412                                  CALL    DSABLE              ; DISABLE ERROR RECOVERY
        022412  020000  005046                  ^O20000,DSABLE
    231 022414                                  BR      JMPRET              ; RETURN TO SEQUENCER
        022414  000000  003746                  ^O0,JMPRET
    232                                         .DSABL  LSB
```

```
      1
      2 022416  000000                .SBTTL  ASSOCIATED VARIABLES FOR GETTING THE HDA SERIAL NUMBER
      3 022417  000000        SUNIT:  .WORD   0                       ; THE ACTIVE UNIT NUMBER
      4 022420                SERRTY: .WORD   0                       ; RETRY COUNT
      5                       SERCHN:
      9 022420  000000                ; THE CHAIN FOR READING SECTOR 0 (RCT)
     13 022421  000000                .WORD   0
     14 022422  023057                .WORD   0
     15 022423  000000                .WORD   SERSEC                  ; POINTER TO THE SECTOR
     16 022424  000000                .WORD   0
     17 022425  000000                .WORD   0
     18 022426  000000                .WORD   0
```

```
                                .SBTTL  ********** OVERLAY MODULE INITD - INITILIZE THE DRIVE  PARAMETERS FOR TESTING
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
022427                          INITD:
                                ;
                                ;       BRING DRIVE ONLINE, CLEAR ALL ERRORS (IF ANY) AND CHANGE THE
                                ;       MODE TO THE REQUIRED VALUES
                                ;
                                        .ENABL  LSB
022427  104207  000004                MOV     #4,R0           ; MAXIMUM OF 4 SUBUNITS
022431  104201  000001                MOV     #U.SUBP,R1      ; R1 WILL POINT TO SUBUNIT POINTERS
022433  105051                       ADD     R5,R1           ; R1 POINTS TO SUBUNIT POINTERS
022434  104212                1$:     MOV     (R1)+,R2        ; GET POINTER TO SUBUNIT INFO
022435                                BMI     2$              ; IF UNIT NOT TESTED, BRANCH
022435  070000  022457                ^070000,2$
022437  104123                       MOV     (R2),R3         ; GET SUBUNIT PARAMETERS
022440                                ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
022440  102203  000100                BIT     #SEQSEK,R3      ; SEE IF SEQUENTIAL SEEKS
022442                                BNE     7$              ; IF SO, BRANCH
022442  050000  022450                ^050000,7$
022444  114000  003002                CLR     SCR1            ; FLAG AS RANDOM LBNS
022446                                BR      8$              ; BRANCH
022446  000000  022453                ^00,8$
022450  104200  177777  003002  7$:   MOV     #-1,SCR1        ; FLAG AS SEQUENTIAL LBNS
022453  102203  040000        8$:     BIT     #INITW,R3       ; SEE IF SUBUNIT WILL BE INITALLY WRITTEN
022455                                BNE     3$              ; IF SO, BRANCH
022455  050000  022464                ^050000,3$
022457  117407                2$:     DEC     R0              ; DECREMENT COUNT
022460                                BNE     1$              ; IF NOT ALL SUBUNITS CHECKED, BRANCH
022460  050000  022434                ^050000,1$
022462                                BR      4$              ; BRANCH
022462  000000  022475                ^00,4$
022464  104657  000046        3$:     MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
022466  101207  040000                BIS     #INITW,R0       ; FLAG UNIT AS SUBUNITS GET INITIALLY WRITTEN
022470  100657  000046                MOV     R0,U.PARM(R5)   ; SAVE
022472  101200  100000  003006        BIS     #IWIPRG,M.PARM  ; FLAG MASTER PARAMETERS AS INITIAL WRITE TO BE DONE
022475  104207  000004        4$:     MOV     #4,R0           ; MAXIMUM OF FOUR SUBUNITS
022477  104201  000005                MOV     #U.SUBP+4,R1    ; R1 WILL POINT TO AFTERS SUBUNIT POINTERS
022501  105051                       ADD     R5,R1           ; R1 POINTS TO AFTERS SUBUNIT POINTERS
022502  114003                       CLR     R3              ; INITILIZE SUBUNIT PROTECTION FLAGS
022503  110203                5$:     ROL     R3              ; ROTATE R3
022504  103203  000001                BIC     #1,R3           ; CLEAR LO BIT
022506  104412                       MOV     -(R1),R2        ; GET SUBUNIT POINTER
022507                                BMI     6$              ; IF NO SUBUNIT, BRANCH
022507  070000  022565                ^070000,6$
022511  104122                       MOV     (R2),R2         ; GET SUBUNIT STATUS
022512                                ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
022512  102202  000100                BIT     #SEQSEK,R2      ; SEE IF SEQUENTIAL SEEKS
022514                                BNE     9$              ; IF SO, BRANCH
022514  050000  022524                ^050000,9$
022516  115000  003002                TST     SCR1            ; SEE IF ALL SUBUNITS RANDOM SEEKS
022520                                BNE     10$             ; IF NOT, BRANCH
```

```
                022520  050000  022530                  ^050000,10$
     68         022522                                  BR      11$             ; BRANCH
                022522  000000  022557                  ^00,11$
     69         022524  115000  003002          9$:     TST     SCR1            ; SEE IF ALL SUBUNITS SEQUENTIAL SEEKS
     70         022526                                  BNE     11$             ; IF SO, BRANCH
                022526  050000  022557                  ^050000,11$
     71         022530                          10$:    DEVFTL  56,#SER20       ; REPORT ERROR
                022530  104200  003604  001601                                          MOV     #ER56,HRQ.04
                022533  104200  004047  001602                                          MOV     #SER20,HRQ.05
                022536  104202  047730                                                  MOV     #56!FTLDEV+4000.,R2
                022540  104020  001577                                                  MOV     R2,HRQ.02
                022542  104200  022542  001576                                          MOV     #.,HRQ.01
                022545  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
     72         022550                                  ENDERR  0
                022550  114000  003013                                                  CLR     ERRPOS          ; CLEAR THE POSITION
     73         022552  101200  000002  003006          BIS     #DIE,M.PARM     ; FLAG INITIALIZATION ERROR
     74         022555                                  BR      DRVEXT          ; EXIT
                022555  000000  022604                  ^00,DRVEXT
     75         022557  102202  004000          11$:    BIT     #RONLY,R2       ; SEE IF READ ONLY DRIVE
     76         022561                                  BEQ     6$              ; IF NOT READ ONLY, BRANCH
                022561  010000  022565                  ^010000,6$
     77         022563  101203  000001                  BIS     #1,R3           ; SET READ ONLY BIT
     78         022565  117407                  6$:     DEC     R0              ; DECREMENT COUNT
     79         022566                                  BNE     5$              ; IF COUNT UNEXPIRED, BRANCH
                022566  050000  022503                  ^050000,5$
     80         022570  110203                          ROL     R3              ; ROTATE MASK TO CORRECT POSITION
     81         022571  110203                          ROL     R3              ; ROTATE MASK TO CORRECT POSITION
     82         022572  110203                          ROL     R3              ; ROTATE MASK TO CORRECT POSITION
     83         022573  110203                          ROL     R3              ; ROTATE MASK TO CORRECT POSITION
     84         022574  103203  177417                  BIC     #LBHINB,R3      ; CLEAR UNUSED BITS
     85         022576  100653  000045                  MOV     R3,U.WPRT(R5)   ; SAVE
     86         022600  104207  005134                  MOV     #SETUP,R0       ; SETUP NEXT ROUTINE CALLED
     87         022602  104051                          MOV     R5,R1           ; MAKE R1 NON-ZERO (DEFERRED CALL)
     88         022603  114002                          CLR     R2              ; NO ERRORS
     89         022604                          DRVEXT: BR      JMPRET          ; RETURN TO SEQUENCER
                022604  000000  003746                  ^00,JMPRET
     90
    108         022606                                  .DSABL  LSB
                                                        ONETIM
                                                        .LIST   ME
                                                .SBTTL  UNIT AND SUBUNIT GET CHARACTERISTICS AND RUN SDI COMMANDS
                                                ;
                                                ;       NOTE: BECAUSE OF THEIR POSITION, THE FOLLOWING SDI LEVEL 2 COMMANDS
                                                ;       WILL BE ISSUED WITH A LONG TIMEOUT
                                                ;
                        000000                  SDI$    =       0               ; DOUBLE UP ON THESE COMMANDS, SINCE IF
                                                                                ; AN ERROR OCCURS, IT'S A DEVICE FATAL
                022606                          CR.GCR: MSG GCR,1,11.,CHRRES    ; GET CHARACTERISTICS
                022606  022626                          .WORD   GCR                     ; ADDRESS OF COMMAND
                022607  000001                          .WORD   1                       ; SIZE OF COMMAND IN BYTES
                022610  000013                          .WORD   11.                     ; SIZE OF REPLY IN WORDS
                022611  000170                          .WORD   CHRRES                  ; SUCCESSFUL COMPLETION CODE
                022612  000035                          .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
                        000001                  SDI$    =       SDI$+1
                022613                          CR.SCR: MSG SCR,2,19.,SBCRES    ; GET SUBUNIT CHARACTERISTICS
                022613  022627                          .WORD   SCR                     ; ADDRESS OF COMMAND
                022614  000002                          .WORD   2                       ; SIZE OF COMMAND IN BYTES
                022615  000023                          .WORD   19.                     ; SIZE OF REPLY IN WORDS
```

```
022616  000167                          .WORD   SBCRES                  ; SUCCESSFUL COMPLETION CODE
022617  000036                          .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
        000002                  SDI$    =       SDI$+1
022620                          CR.RUN: MSG RUN,1,7,COMPLT      ; INITIATE LOAD
022620  022625                          .WORD   RUN                     ; ADDRESS OF COMMAND
022621  000001                          .WORD   1                       ; SIZE OF COMMAND IN BYTES
022622  000007                          .WORD   7                       ; SIZE OF REPLY IN WORDS
022623  000176                          .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
022624  000037                          .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
        000003                  SDI$    =       SDI$+1
022625     000      014         RUN:    .BYTE   0,DRVRUN        ; DRIVE RUN
022626     000      207         GCR:    .BYTE   0,GETCHR        ; GET CHARACTERISTICS WITH A
022627     000      210         SCR:    .BYTE   0,GETSUB        ; GET SUBUNIT CHARACTERISTICS
022630  000000                  SUBUNT: .WORD   0               ; SUBUNIT SELECTION IN LOW ORDER BYTE
                                ;
                                ;       CHECK THAT 4 UNITS WITH 8 SUBUNITS IN ALL, WITH 16 BAD BLOCKS
                                ;       SPECIFIED EACH, CAN FIT IN MEMORY WITHOUT OVERLAPPING WITH
                                ;       THE OVERLAYS OR THE R/W BUFFER SPACE
                                ;
                                .SBTTL  ********** NON-OVERLAY MODULE START - TEST 4 INITIALIZATION
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;       THIS 'OVERLAY' IS LOADED ALONG WITH THE INITIAL DOWNLINE LOAD OF
                                ;       TEST 4.  ONCE EXECUTED, THIS OVERLAY IS NEVER LOADED AGAIN UNLESS
                                ;       THE ENTIRE TEST IS STARTED (OR RESTARTED) AGAIN.  ALL OTHER
                                ;       OVERLAYS ARE LOADED INTO THE SPACE THAT THIS ONE OCCUPIES INITIALLY,
                                ;       THUS DESTROYING THIS OVERLAY.
                                ;
                                ;
022631                          UCURSR: .BLKW   1
022632  037705                  LASTU:  .WORD   FIRSTU
022633                          UMASK:  .BLKW   1
022634                          NUMBB:  .BLKW   1
022635                          MAXDBN: .BLKW   2
022637                          SECTRK: .BLKW   1
022640                          SECGRP: .BLKW   1
022641                          SECCYL: .BLKW   1
022642                          TGS:    .BLKW   7
022651                          DSERNM: .BLKW   3               ; TEMP STORAGE FOR DRIVE SERIAL NUMBER
022654                          START:  CALL    GMPARM          ; GET MASTER PARAMETERS
022654  020000  023056                  ^O020000,GMPARM
                                        .IF B GMPARM
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
022656                                  CALL    GETU            ; GET ALL UNITS TO TEST
022656  020000  023101                  ^O020000,GETU
                                        .IF B GETU
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
022660                                  CALL    BLDUNT          ; BUILD UNIT AND SUBUNIT PARAMETERS
022660  020000  023573                  ^O020000,BLDUNT
                                        .IF B BLDUNT
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
```

```
                                        .ENDC
022662  101200  000001  003006          BIS     #INTINP,M.PARM  ; MARK AS INITIALIZATION IN PROGRESS
022665                                  CALL    TROOT           ; FILL OUT UNIT AND SUBUNIT PARM'S, GET SUBUNIT CHAR
022665  020000  023012                  ^O020000,TROOT
                                        .IF B TROOT
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
022667  102200  000002  003006          BIT     #DIE,M.PARM     ; SEE IF INITIALIZATION ERRORS
022672                                  BEQ     3$              ; IF NOT, BRANCH
022672  010000  022725                  ^O010000,3$
                                        .IF B 3$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
022674                                  DEVFTL  16,#SER39       ; REPORT INITIALIZATION ERRORS
022674                                  ERROR   FTLDEV,16,<#SER39>
        000012                          .RADIX  10
        000005                          NUMPTR  =       5
022674                                  MOVMSG  #ER16,4

                                        .IF     LT,4-10
022674  104200  001553  001601                                          MOV     #ER16,HRQ.04
                                        .IFF
                                                                        MOV     #ER16,HRQ.4
                                        .ENDC
                                        .IF     NB,<#SER39>
                                        .IRP    X,<#SER39>
                                        MOVMSG  X,\NUMPTR
                                        NUMPTR  =       NUMPTR + 1
                                        .ENDR
022677                                  MOVMSG  #SER39,\NUMPTR

                                        .IF     LT,5-10
022677  104200  002477  001602                                          MOV     #SER39,HRQ.05
                                        .IFF
                                                                        MOV     #SER39,HRQ.5
                                        .ENDC
        000006                          NUMPTR  =       NUMPTR + 1
                                        .ENDC
022702  104202  047660                                                  MOV     #16!FTLDEV+4000.,R2
022704  104020  001577                                                  MOV     R2,HRQ.02
022706  104200  022706  001576                                          MOV     #.,HRQ.01
        000010                          .RADIX
022711  104200  060014  001575                                          MOV     #ERRMC,HRQ.RQ
022714  104200  177777  001600          MOV     #-1,HRQ.03      ; MOVE 'NOT ASSICIATED WITH ANY UNIT' TO UNIT #
022717  104307  001575                  MOV     HRQ.RQ,R0       ; GET REQUEST
022721                                  CALL    HOSTRQ          ; REPORT
022721  020000  001543                  ^O020000,HOSTRQ
                                        .IF B HOSTRQ
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
022723                                  BR      STOP            ; STOP TEST 4
022723  000000  003370                  ^O00,STOP
                                        .IF B STOP
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
022725  103200  000001  003006  3$:     BIC     #INTINP,M.PARM  ; INITIALIZATION NO LONGER IN PROGRESS
022730  104307  003012                  MOV     MEMPOL,R0       ; R0 HAS POINTER TO FREE MEMORY
```

```
022732  107207  017005                  SUB     #BUFARA,R0      ; SUBTRACT END OF CODE FROM FREE MEMORY
022734  104073                          MOV     R0,R3           ; SAVE AMOUNT OF FREE MEMORY
022735  114001                          CLR     R1              ; CLEAR COUNT
022736  107207  000425          1$:     SUB     #RBUFLN+LINKLN,R0       ; SUBTRACT NEEDED MEM FOR READ
022740                                  BMI     2$              ; IF NO MORE MEMORY, BRANCH
022740  070000  022745                  ^070000,2$
                                        .IF B 2$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
022742  115401                          INC     R1              ; INCREMENT COUNT
022743                                  BR      1$              ; LOOP
022743  000000  022736                  ^00,1$
                                        .IF B 1$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
022745  104010  003016          2$:     MOV     R1,SECMAX       ; SAVE IN SECTOR MAXIMUM
022747  114001                          CLR     R1              ; CLEAR COUNT
022750  107203  000411                  SUB     #WBUFLN+LINKLN,R3       ; SUBTRACT NEEDED MEM FOR WRITE
022752  115401                  4$:     INC     R1              ; INCREMENT COUNT
022753  107203  000010                  SUB     #LINKLN,R3      ; SUBTRACT LINK LENGTH FROM MEMORY
022755                                  BMI     5$              ; IF MEMORY EXHAUSTED, BRANCH
022755  070000  022761                  ^070000,5$
                                        .IF B 5$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
022757                                  BR      4$              ; LOOP
022757  000000  022752                  ^00,4$
                                        .IF B 4$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
022761  104010  003017          5$:     MOV     R1,CHNMAX       ; SAVE IN SECTOR MAXIMUM
022763                                  BR      ROOT            ; START EXERCISE
022763  000000  003315                  ^00,ROOT
                                        .IF B ROOT
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
                                .SBTTL  GETMEM - ALLOCATES MEMORY FOR UNIT AND SUBUNIT BLOCK AND SUBUNIT PARAMETERS
022765                          GETMEM:
                                ;
                                ;       RETURN A BLOCK OF MEMORY
                                ;
022765  107070  003012                  SUB     R0,MEMPOL       ; SUBTRACT REQUESTED LENGTH FROM MEMOY POOL
022767  104307  003012                  MOV     MEMPOL,R0       ; LOAD R0 WITH POINTER TO REQUESTED MEMORY
022771                                  RETURN                  ; RETURN TO CALLING PROGRAM
022771                                  JMP     0
022771  000000  000000                  ^00,0
                                        .IF B 0
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
                                .SBTTL  CBB2    - 28 BIT COMPARE FOR SETUP MODULES
022773                          CBB2:
                                ;
                                ;       28 BIT COMPARE FOR BAD BLOCKS
                                ;
022773  104634  000001                  MOV     1(R3),R4        ; GET WORD THAT R3 POINTS TO
022775  103204  170000                  BIC     #^CHBHINB,R4    ; STRIP OFF UNUSED BITS
022777  104625  000001                  MOV     1(R2),R5        ; GET OTHER WORD TO COMPARE
```

```
023001  103205  170000                  BIC     #^CHBHINB,R5    ; STRIP OFF THE UNUSED BITS
023003  106045                          CMP     R4,R5           ; COMPARE
023004                                  BNE     1$              ; IF DIFFERENCE IS FOUND, BRANCH
023004  050000  023010                  ^O50000,1$
                                        .IF B 1$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023006  104125                          MOV     (R2),R5         ; GET OTHER WORD TO COMPARE
023007  106135                          CMP     (R3),R5         ; COMPARE
023010                          1$:     RETURN                  ; RETURN TO SORTBE
023010                                  JMP     0
023010  000000  000000                  ^O0,0
                                        .IF B 0
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
                                .SBTTL  TROOT  - TEMPORARY ROOT FOR SEQNCR DURING TEST 4 SETUP
023012                          TROOT:
                                ;
                                ;       TROOT WILL FILL OUT THE AS YET UNDEFINED FIELDS IN BOTH UNIT
                                ;       AND SUBUNIT PARAMETERS, AS WELL AS GETTING THE SUBUNIT CAHRACTERISTICS
                                ;
023012  104205  037705                  MOV     #FIRSTU,R5      ; R5 POINTS TO FIRST SUBUNIT
023014  114002                  MKLOOP: CLR     R2              ; FOR ZEROING RECOVERY WORD
023015  100652  000047                  MOV     R2,U.RCOV(R5)   ; ZERO RECOVERY WORD
023017  104207  001000                  MOV     #FTIME,R0       ; MARK AS FIRST TIME (WILL DISABLE RECOVERY)
023021  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE
023023  104201  017005                  MOV     #INSET,R1       ; FIRST MODULE IS INSET
023025  100651  000013                  MOV     R1,U.NFUN(R5)   ; SAVE IN UNIT PARAMETERS
023027  104201  000003                  MOV     #3,R1           ; SDI SHORT TIMEOUT SET TO 30 SEC
023031  100651  000033                  MOV     R1,U.SDIS(R5)   ; MOVE TO PARAMETERS
023033  104201  000360                  MOV     #360,R1         ; WRITE PROTECT THE ENTIRE DRIVE
023035  100651  000045                  MOV     R1,U.WPRT(R5)   ; SAVE
023037                                  CALL    SEQNCR          ; RUN SEQUENCER
023037  020000  003375                  ^O20000,SEQNCR
                                        .IF B SEQNCR
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023041  104657  000046                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
023043  101207  001000                  BIS     #FTIME,R0       ; SET FIRST TIME
023045  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE
023047  104155                          MOV     (R5),R5         ; TRAVERSE TO NEXT UNIT
023050                                  ASSUME  U.NEXT,0        ; ASSUME U.NEXT OFFSET IS ZERO
                                        .IF     NE,U.NEXT-0
                                        .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                        .ENDC
023050  106205  037705                  CMP     #FIRSTU,R5      ; SEE IF TRAVERSED ENTIRE RING
023052                                  BNE     MKLOOP          ; IF NOT, BRANCH
023052  050000  023014                  ^O50000,MKLOOP
                                        .IF B MKLOOP
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023054                                  RETURN                  ; RETURN TO CALLING PROGRAM
023054                                  JMP     0
023054  000000  000000                  ^O0,0
                                        .IF B 0
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
```

```
                023057                  SERSEC  =       .+1             ; SERIAL NUMBER SECTOR AREA
                                        .SBTTL  GMPARM - GET MASTER PARAMETERS (PATTERN 16) AND SET UP OVERLAY ADDRESSES
        023056                          GMPARM:
                                        ;
                                        ;       GET MASTER PARAMETERS
                                        ;
        023056  104207  060003                 MOV     #T4MPRM,R0      ; R0 CONTAINS HOST REQUEST
        023060                                  CALL    HOSTRQ          ; INITIATE HOST REQUEST
        023060  020000  001543                  ^O20000,HOSTRQ
                                                .IF B HOSTRQ
                                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                                .ENDC
        023062  104207  001576                  MOV     #HRQ.01,R0      ; R0 POINTS TO PATTERN INFORMATION
        023064  104272                          MOV     (R0)+,R2        ; R2 HAS LENGTH OF PATTERN
        023065                                  BEQ     3$              ; IF NO PATTERN, BRANCH
        023065  010000  023077                  ^O10000,3$
                                                .IF B 3$
                                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                                .ENDC
        023067  104201  003042                  MOV     #PAT0+1,R1      ; R1 POINTS TO PATTERN 16 AREA (SKIP EDC)
        023071  100212                          MOV     R2,(R1)+        ; MOVE PATTERN LENGTH TO PAT0 AREA
        023072  104273          2$:             MOV     (R0)+,R3        ; GET WORD OF PATTERN
        023073  100213                          MOV     R3,(R1)+        ; MOVE TO PATTERN 16 AREA
        023074  117402                          DEC     R2              ; DECREMENT COUNT
        023075                                  BNE     2$              ; IF COUNT UNEXHAUSTED, BRANCH
        023075  050000  023072                  ^O50000,2$
                                                .IF B 2$
                                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                                .ENDC
        023077                  3$:             RETURN                  ; RETURN TO CALLING PROGRAM
        023077                                  JMP     0
        023077  000000  000000                  ^O0,0
                                                .IF B 0
                                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                                .ENDC
                                        .SBTTL  GETU   - POLL ALL PORTS, THEN GET UNITS TO TEST
        023101                          GETU:
                                        ;
                                        ;       GET THE UNITS TO TEST
                                        ;
                                        ;
                                        ;
                                        ;       POLL ALL PORTS AND FILL IN A UDA PORT INFORMATION TABLE (UNITS)
                                        ;
        023101  104205  000001                  MOV     #1,R5           ; MOVE INITIAL MASK TO R5
        023103  104204  023553                  MOV     #UNITS,R4       ; R4 POINTS TO UNIT TABLE
        023105                  5$:             PUSH    R4              ; SAVE R4
                                                .IRP X,<R4>
                                                                                        MOV X,-(SP)
                                                .ENDR
        023105  100464                                                  ;               MOV R4,-(SP)
        023106  104052                          MOV     R5,R2           ; MOVE MASK TO R2
        023107                                  CALL    RDSTAT          ; GET DRIVE'S STATUS
        023107  020000  001470                  ^O20000,RDSTAT
                                                .IF B RDSTAT
                                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                                .ENDC
        023111  115002                          TST     R2              ; SEE IF ERROR
```

```
023112                                  BEQ     20$             ; IF NOT, BRANCH
023112  010000  023130                  ^010000,20$
                                        .IF B 20$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023114                                  BMI     10$             ; IF MSB, PARITY ERRORS > HALF SECOND
023114  070000  023122                  ^070000,10$
                                        .IF B 10$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023116  104203  005205                  MOV     #SER10,R3       ; NO DRIVE ATTACHED
023120                                  BR      15$             ; BRANCH
023120  000000  023124                  ^00,15$
                                        .IF B 15$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023122  104203  002243          10$:    MOV     #SER45,R3       ; PARITY ERRORS > HALF SECOND
023124  100643  000001          15$:    MOV     R3,1(R4)        ; SAVE ERROR MESSAGE
023126                                  BR      100$            ; REPORT
023126  000000  023353                  ^00,100$
                                        .IF B 100$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023130  114003                  20$:    CLR     R3              ; SET UP TIMEOUT COUNT
023131  104052                  25$:    MOV     R5,R2           ; MOVE PORT SELECT TO R2
023132                                  CALL    RDSTAT          ; GET STATUS
023132  020000  001470                  ^020000,RDSTAT
                                        .IF B RDSTAT
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023134  115002                          TST     R2              ; SEE IF ERROR
023135                                  BNE     30$             ; IF SO, BRANCH
023135  050000  023143                  ^050000,30$
                                        .IF B 30$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023137  102201  000001                  BIT     #RCVRDY,R1      ; SEE IF RECEIVER READY ASSERTED
023141                                  BNE     35$             ; IF SO, BRANCH
023141  050000  023154                  ^050000,35$
                                        .IF B 35$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023143  117403                  30$:    DEC     R3              ; DECREMENT COUNT
023144                                  BNE     25$             ; IF INCOMPLETE, BRANCH
023144  050000  023131                  ^050000,25$
                                        .IF B 25$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023146  104203  005220                  MOV     #SER11,R3       ; RECEIVER READY NEVER ASSERTED
023150  100643  000001                  MOV     R3,1(R4)        ; SAVE ERROR MESSAGE
023152                                  BR      100$            ; REPORT
023152  000000  023353                  ^00,100$
                                        .IF B 100$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023154  102201  000100          35$.    BIT     #AVAIL,R1       ; SEE IF DRIVE IS AVAILABLE
023156                                  BNE     40$             ; IF SO, BRANCH
```

```
GETU   - POLL ALL PORTS, THEN GET UNITS TO TEST

        023156  050000  023166                 ^050000,40$
                                                .IF B 40$
                                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                                .ENDC
        023160  104203  005435                 MOV     #SER40,R3       ; GET SECONDARY ERROR
        023162  100643  000001                 MOV     R3,1(R4)        ; SAVE
        023164                                  BR      100$            ; EXIT
        023164  000000  023353                 ^00,100$
                                                .IF B 100$
                                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                                .ENDC
        023166  104202  001750          40$:    MOV     #MAXSND,R2      ; SET UP MAXIMUM TRIES AT SENDING
        023170  104203  002357                  MOV     #CR.GST,R3      ; R3 POINTS TO GET STATUS COMMAND
        023172  104137          45$:    MOV     (R3),R0         ; SET ADR OF SDI COMMAND BUFFER
        023173                          ASSUME  L2.OPC,0
                                        .IF     NE,L2.OPC-0
                                        .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                        .ENDC
        023173  104631  000001                 MOV     L2.SLN(R3),R1   ; SET BUFFER LENGTH
        023175                                  PUSH    R2              ; SAVE R2
                                                .IRP X,<R2>
                                                                                MOV X,-(SP)
                                                .ENDR
        023175  100462                                                          MOV R2,-(SP)
        023176  104052                          MOV     R5,R2           ; SETUP FOR SEND
        023177  060004                          XFC     SEND            ; SEND COMMAND
        023200                                  POP     R2              ; RESTORE COUNT
                                                .IRP X,<R2>
                                                                                MOV (SP)+,X
                                                .ENDR
        023200  104262                                                          MOV (SP)+,R2
        023201  115001                          TST     R1              ; DID UNIT ACCEPT COMMAND
        023202                                  BEQ     50$             ; IF SO, BRANCH
        023202  010000  023215                 ^010000,50$
                                                .IF B 50$
                                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                                .ENDC
        023204  117402                          DEC     R2              ; DECREMENT COUNT
        023205                                  BNE     45$             ; IF UNEXPIRED, BRANCH
        023205  050000  023172                 ^050000,45$
                                                .IF B 45$
                                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                                .ENDC
        023207  104203  005236                  MOV     #SER12,R3       ; GET ERROR NUMBER
        023211  100643  000001                  MOV     R3,1(R4)        ; SAVE
        023213                                  BR      100$
        023213  000000  023353                 ^00,100$
                                                .IF B 100$
                                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                                .ENDC
        023215                  50$:    PUSH    R4              ; SAVE R4
                                        .IRP X,<R4>
                                                                                MOV X,-(SP)
                                        .ENDR
        023215  100464                                                          MOV R4,-(SP)
        023216  104204  000003                  MOV     #3,R4           ; SET UP SHORT TIMEOUT
        023220  104207  002442          55$:    MOV     #ST,R0          ; SET DATA BUFFER ADDRESS
```

GETU - POLL ALL PORTS, THEN GET UNITS TO TEST

```
023222  104631  000002                  MOV     L2.RLN(R3),R1   ; SET BUFFER LENGTH
023224  104052                          MOV     R5,R2           ; SETUP FOR RECEIVE
023225  060005                          XFC     RCV             ; RECEIVE SDI COMMAND
023226  115001                          TST     R1              ; DID ERROR OCCUR
023227                                  BEQ     85$             ; IF NOT, BRANCH
023227  010000  023302                  ^010000,85$
                                        .IF B 85$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023231  106201  000001                  CMP     #1,R1           ; SEE IF TIMEOUT
023233                                  BNE     60$             ; IF NOT, BRANCH
023233  050000  023245                  ^050000,60$
                                        .IF B 60$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023235  117404                          DEC     R4              ; DECREMENT TIMEOUT VALUE
023236                                  BNE     55$             ; IF NOT TIMEOUT, BRANCH
023236  050000  023220                  ^050000,55$
                                        .IF B 55$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023240                                  POP     R4              ; RESTORE R4
                                        .IRP X,<R4>
                                                                                MOV (SP)+,X
                                        .ENDR
                                                                                MOV (SP)+,R4
023240  104264
023241  104203  005250                  MOV     #SER13,R3       ; GET ERROR NUMBER
023243                                  BR      80$             ; BRANCH TO EXIT
023243  000000  023276                  ^00,80$
                                        .IF B 80$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023245                          60$:    POP     R4              ; RESTORE R4
                                        .IRP X,<R4>
                                                                                MOV (SP)+,X
                                        .ENDR
                                                                                MOV (SP)+,R4
023245  104264
023246  110601                          ROR     R1              ; ROTATE INTO POSITION TO TEST
023247  110601                          ROR     R1              ; SEE IF FIRST WORD NOT START FRAME
023250                                  BCC     65$             ; IF NOT, BRANCH
023250  040000  023256                  ^040000,65$
                                        .IF B 65$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023252  104203  005263                  MOV     #SER14,R3       ; GET ERROR NUMBER
023254                                  BR      80$             ; BRANCH TO END OF LOOP
023254  000000  023276                  ^00,80$
                                        .IF B 80$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023256  110601                  65$:    ROR     R1              ; SEE IF FRAMING ERROR
023257                                  BCC     70$             ; IF NOT, BRANCH
023257  040000  023265                  ^040000,70$
                                        .IF B 70$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023261  104203  005311                  MOV     #SER15,R3       ; GET ERROR NUMBER
```

GETU - POLL ALL PORTS, THEN GET UNITS TO TEST

```
023263                                  BR      80$             ; BRANCH TO END OF LOOP
023263  000000  023276                  ^O00,80$
                                        .IF B 80$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023265  110601                  70$:    MOV     R1              ; SEE IF CHECKSUM ERROR
023266                                  BCC     75$             ; IF NOT, BRANCH
023266  040000  023274                  ^O040000,75$
                                        .IF B 75$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023270  104203  005333                  MOV     #SER16,R3       ; GET ERROR NUMBER
023272                                  BR      80$             ; BRANCH TO END OF LOOP
023272  000000  023276                  ^O00,80$
                                        .IF B 80$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023274  104203  005356          75$:    MOV     #SER17,R3       ; GET ERROR NUMBER
023276  100643  000001          80$:    MOV     R3,1(R4)        ; SAVE
023300                                  BR      100$            ; BRANCH TO END OF LOOP
023300  000000  023353                  ^O00,100$
                                        .IF B 100$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
                                ;
                                ;       NOW FILL IN THE TABLE WITH ALL THE SUBUNIT NUMBERS
                                ;
023302                          85$:    POP     R4              ; RESTORE R4
                                        .IRP X,<R4>
                                                                                MOV (SP)+,X
                                        .ENDR
                                                                                MOV (SP)+,R4
023302  104264                  MOV     #-1,R0          ; GET 'NO UNITS' FLAG
023303  104207  177777          MOV     R0,1(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
023305  100647  000001          MOV     R0,2(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
023307  100647  000002          MOV     ST,R0           ; R0 HAS UNIT NUMBER
023311  104307  002442          MOV     R0,R2           ; COPY R0 TO R2
023313  104072                  BIC     #^CHBHINB,R0    ; R0 HAS UNIT NUMBER
023314  103207  170000          PUSH    <R1,R2>         ; SAVE R1 AND R2
023316                          .IRP X,<R1,R2>
                                                                                MOV X,-(SP)
                                        .ENDR
                                                                                MOV R1,-(SP)
023316  100461                                                                  MOV R2,-(SP)
023317  100462
023320  104052                  MOV     R5,R2           ; MOVE UDA PORT MASK TO R2
023321                          CALL    RDSTAT          ; GET STATUS
023321  020000  001470          ^O020000,RDSTAT
                                .IF B RDSTAT
                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                .ENDC
023323  102201  000002          BIT     #ATTN,R1        ; SEE IF SPINABLE
023325                          BNE     90$             ; IF SO, BRANCH
023325  050000  023331          ^O050000,90$
                                .IF B 90$
                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                .ENDC
023327  101207  010000          BIS     #10000,R0       ; SET 'NOT SPINABLE' FLAG
```

GETU - POLL ALL PORTS, THEN GET UNITS TO TEST

```
023331                          90$:    POP     <R2,R1>         ; RESTORE
                                        .IRP X,<R2,R1>
                                                                                MOV (SP)+,X
                                        .ENDR
023331  104262                                                                  MOV (SP)+,R2
023332  104261                                                                  MOV (SP)+,R1
023333  101207  040000                 BIS     #40000,R0       ; FLAG UNIT AS NOT TESTED
023335  110702                          SWAB    R2              ; SWAP R2'S BYTES
023336  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
023337  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
023340  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
023341  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
023342  103202  177760                 BIC     #LBLONB,R2      ; CLEAR ALL BUT SUBUNIT BITS
023344  110602                  95$:    ROR     R2              ; MOVE SUBUNIT BIT TO CARRY
023345                                  BCC     100$            ; IF NO MORE SUBUNITS, BRANCH
023345  040000  023353                 ^040000,100$
                                        .IF B 100$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023347  100247                          MOV     R0,(R4)+        ; MOVE SUBUNIT NUMBER TO TABLE
023350  115407                          INC     R0              ; INCREMENT SUBUNIT NUMBER
023351                                  BR      95$             ; BRANCH
023351  000000  023344                 ^00,95$
                                        .IF B 95$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC

023353                          100$:   POP     R4              ; R4 POINTS TO START OF UNIT JUST HANDLED
                                        .IRP X,<R4>
                                                                                MOV (SP)+,X
                                        .ENDR
023353  104264                                                                  MOV (SP)+,R4
023354  105204  000004                 ADD     #4,R4           ; R4 WILL POINT TO NEXT UNIT (CLEAR CARRY FOR ROL)
023356  110205                          ROL     R5              ; R5 HAS NEXT UNIT PORT MASK
023357  106205  000020                 CMP     #20,R5          ; SEE IF ALL PORTS TESTED
023361                                  BNE     5$              ; IF NOT, BRANCH
023361  050000  023105                 ^050000,5$
                                        .IF B 5$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
                                ;
                                ;       NOW GET THE PLUG NUMBERS TO TEST AND FIND THEM IN THE TABLE
                                ;
023363  104207  060012                 MOV     #UTOTST,R0      ; GET WHAT SUBUNIT NUMBERS TO TEST REQUEST
023365                                  CALL    HOSTRQ          ; GET THE PLUG NUMBERS
023365  020000  001543                 ^020000,HOSTRQ
                                        .IF B HOSTRQ
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023367  104207  001576                 MOV     #HRQ.01,R0      ; R0 POINTS TO UNIT NUMBERS TO TEST
023371  104201  023553          105$:  MOV     #UNITS,R1       ; R1 POINTS TO UDA PORT INFORMATION
023373  104112                  110$:  MOV     (R1),R2         ; R2 HAS UNIT
023374                                  BMI     130$            ; IF NONE, BRANCH
023374  070000  023424                 ^070000,130$
                                        .IF B 130$
```

```
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023376                                  PUSH    R1              ; SAVE R1
                                        .IRP X,<R1>
                                                                                MOV X,-(SP)
                                        .ENDR
023376  100461                                                                  MOV R1,-(SP)
023377  103202  160000          115$:   BIC     #160000,R2      ; CLEAR 'NOT TESTED', LEAVE 'UNSPINABLE' SET
023401  106172                  ;       CMP     (R0),R2         ; SEE IF IT IS A UNIT TO TEST
023402                          ;       BNE     120$            ; NO MATCH
023402  050000  023410          ;       ^050000,120$
                                ;       .IF B 120$
                                ;       .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                ;       .ENDC
023404  100112                  ;       MOV     R2,(R1)         ; SAVE UNIT AS ONE TO TEST
023405                          ;       POP     R1              ; RESTORE STACK
                                ;       .IRP X,<R1>
                                                                                MOV (SP)+,X
                                ;       .ENDR
023405  104261                                                                  MOV (SP)+,R1
023406                          ;       BR      175$            ; LOOK FOR THE NEXT ONE
023406  000000  023541          ;       ^00,175$
                                ;       .IF B 175$
                                ;       .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                ;       .ENDC
023410  115401          120$:   INC     R1              ; POINT TO NEXT SUBUNIT
023411  104012                  MOV     R1,R2           ; COPY TO R2
023412  107202  023553          SUB     #UNITS,R2       ; SUBTRACT STARTING ADDRESS
023414  102202  000003          BIT     #3,R2           ; SEE IF STILL ON SAME UNIT
023416                          BEQ     125$            ; IF NOT, BRANCH
023416  010000  023423          ^010000,125$
                                .IF B 125$
                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                .ENDC
023420  104112                  MOV     (R1),R2         ; SEE IF ANY MORE SUBUNITS
023421                          BPL     115$            ; IF SO, BRANCH
023421  030000  023377          ^030000,115$
                                .IF B 115$
                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                .ENDC
023423                  125$:   POP     R1              ; RESTORE R1
                                .IRP X,<R1>
                                                                                MOV (SP)+,X
                                .ENDR
023423  104261                                                                  MOV (SP)+,R1
023424  105201  000004  130$:   ADD     #4,R1           ; LOOK AT NEXT UNIT
023426  106201  023573          CMP     #UNITS+16.,R1   ; SEE IF ENTIRE TABLE SEARCHED
023430                          BNE     110$            ; IF NOT, BRANCH
023430  050000  023373          ^050000,110$
                                .IF B 110$
                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                .ENDC
                        ;
                        ;       DIDN'T FIND THE REQUESTED UNITS -- DUMP ALL KNOWLEDGE OF THE
                        ;       UDA PORTS AND DIE
                        ;
023432  104170  001600          MOV     (R0),HRQ.03     ; SAVE UNIT NUMBER IN REQUEST BUFFER
```

GETU - POLL ALL PORTS, THEN GET UNITS TO TEST

```
023434  104204  001602                  MOV     #HRQ.05,R4      ; R4 POINTS TO OUTPUT BUFFER
023436  104205  023553                  MOV     #UNITS,R5       ; R5 POINTS TO UNIT TABLE
023440  104157                  135$:   MOV     (R5),R0         ; GET FIRST WORD OF UNIT
023441                                  BPL     140$            ; IF VALID UNIT WAS FOUND, BRANCH
023441  030000  023450                  ^O030000,140$
                                        .IF B 140$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023443  104657  000001                  MOV     1(R5),R0        ; GET POINTER TO ERROR MESSAGE
023445  100247                          MOV     R0,(R4)+        ; SAVE IN OUTPUT BUFFER
023446                                  BR      170$            ; EXIT
023446  000000  023510                  ^O00,170$
                                        .IF B 170$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023450                          140$:   PUSH    R5              ; SAVE POINTER TO UNIT TABLE
                                        .IRP X,<R5>
                                                                                MOV X,-(SP)
                                        .ENDR
023450  100465                                                                  MOV R5,-(SP)
023451  102207  010000                  BIT     #10000,R0       ; SEE IF DRIVE UNSPINABLE
023453                                  BEQ     145$            ; IF SPINABLE, BRANCH
023453  010000  023461                  ^O010000,145$
                                        .IF B 145$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023455  104207  005457                  MOV     #SER41,R0       ; REPORT DRIVE(S) UNSPINABLE
023457                                  BR      150$            ; BRANCH
023457  000000  023463                  ^O00,150$
                                        .IF B 150$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023461  104207  005410          145$:   MOV     #SER18,R0       ; GET POINTER TO ERROR MESSAGE
023463  100247          150$:   MOV     R0,(R4)+        ; SAVE
023464  114007                  CLR     R0              ; CLEAR COUNT
023465  104251          155$:   MOV     (R5)+,R1        ; GET UNIT NUMBER
023466                                  BMI     160$            ; IF INVALID, BRANCH
023466  070000  023475                  ^O070000,160$
                                        .IF B 160$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023470  115407                          INC     R0              ; INCREMENT COUNT
023471  106207  000004                  CMP     #4,R0           ; SEE IF MAX
023473                                  BNE     155$            ; IF NOT, LOOP
023473  050000  023465                  ^O050000,155$
                                        .IF B 155$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023475  104671  023546          160$:   MOV     SER18E-1(R0),R1 ; GET POINTER TO CORRECT ERROR MESSAGE
023477  100241                          MOV     R1,(R4)+        ; MOVE INTO OUTPUT BUFFER
023500                                  POP     R5              ; RESTORE R5
                                        .IRP X,<R5>
                                                                                MOV (SP)+,X
                                        .ENDR
023500  104265                                                                  MOV (SP)+,R5
023501                                  PUSH    R5              ; SAVE R5
                                        .IRP X,<R5>
```

GETU - POLL ALL PORTS, THEN GET UNITS TO TEST

```
                                                                                                MOV X,-(SP)
                                        .ENDR
                                                                                                MOV R5,-(SP)
023501  100465
023502  104251                  165$:   MOV     (R5)+,R1        ; GET SUBUNIT NUMBER
023503  100241                          MOV     R1,(R4)+        ; SAVE
023504  117407                          DEC     R0              ; DECREMENT COUNT
023505                                  BNE     165$            ; IF COUNT INCOMPLETE, BRANCH
023505  050000  023502                  ^O050000,165$
                                        .IF B 165$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023507                                  POP     R5              ; RESTORE R5
                                        .IRP X,<R5>
                                                                                                MOV (SP)+,X
                                        .ENDR
023507  104265                                                                                  MOV (SP)+,R5
023510  105205  000004          170$:   ADD     #4,R5           ; POINT TO NEXT UNIT TABLE
023512  106205  023573                  CMP     #UNITS+16.,R5   ; SEE IF ENTIRE TABLE SEARCHED
023514                                  BNE     135$            ; IF NOT, BRANCH
023514  050000  023440                  ^O050000,135$
                                        .IF B 135$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023516                                  DEVFTL  1000            ; REPORT FATAL ERROR (WILL SHOW UP AS A 5000)
023516                                  ERROR   FTLDEV,1000,<>
        000012                          .RADIX  10
        000005                          NUMPTR  =       5
023516                                  MOVMSG  #ER1000,4

                                        .IF     LT,4-10
023516  104200  005066  001601                                                MOV     #ER1000,HRQ.04
                                        .IFF
                                                                                MOV     #ER1000,HRQ.4
                                        .ENDC
                                        .IF     NB,<>
                                        .IRP    X,<>
                                        MOVMSG  X,\NUMPTR
                                        NUMPTR  =       NUMPTR + 1
                                        .ENDR
                                        .ENDC
023521  104202  051610                                                        MOV     #1000!FTLDEV+4000.,R2
023523  104020  001577                                                        MOV     R2,HRQ.02
023525  104200  023525  001576                                               MOV     #.,HRQ.01
        000010                          .RADIX
023530  104200  060014  001575                                               MOV     #ERRMC,HRQ.RQ
023533  104307  001575                  MOV     HRQ.RQ,R0       ; SET UP FOR REPORT
023535                                  CALL    HOSTRQ          ; REPORT ERROR
023535  020000  001543                  ^O020000,HOSTRQ
                                        .IF B HOSTRQ
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023537                                  BR      STOP            ; END TEST
023537  000000  003370                  ^O00,STOP
                                        .IF B STOP
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
```

```
023541  115407                  175$:   INC     R0              ; POINT TO NEXT UNIT TO TEST
023542  104172                          MOV     (R0),R2         ; CHECK NEXT UNIT
023543                                  BPL     105$            ; FIND IN UDA PORT INFORMATION
023543  030000  023371                  ^O030000,105$
                                        .IF B 105$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023545                                  RETURN                  ; RETURN TO INITIALIZATION CODE
023545                                  JMP     0
023545  000000  000000                  ^O0,0
                                        .IF B 0
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC

023547  005432                  SER18E: .WORD   SER18A          ; FOR MULTIPLE SUBUNIT ERROR REPORTING
023550  005426                          .WORD   SER18B
023551  005422                          .WORD   SER18C
023552  005416                          .WORD   SER18D

023553  000020                  UNITS:  .REPT   16.
                                        .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
                                        .ENDR
023553  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023554  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023555  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023556  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023557  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023560  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023561  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023562  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023563  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023564  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023565  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023566  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023567  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023570  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023571  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
023572  177777                          .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
                                .SBTTL  BLDUNT - BUILD THE UNIT PARAMETER BLOCK
023573                          BLDUNT:
                                ;
                                ;       BLDUNT WILL BUILD THE UNIT AND SUBUNIT PARAMETERS
                                ;
023573  104202  000001                  MOV     #1,R2           ; PORT 1 MASK
023575  104020  022633                  MOV     R2,UMASK        ; SAVE MASK
023577  104204  023553                  MOV     #UNITS,R4       ; POINT TO UNITS TO BE TESTED TABLE
023601  104147                  ULOP:   MOV     (R4),R0         ; GET UNIT TO BE TESTED
023602                                  BMI     NOU             ; IF FIRST UNIT NOT TO BE TESTED, BRANCH
023602  070000  023674                  ^O070000,NOU
                                        .IF B NOU
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023604  102207  040000                  BIT     #40000,R0       ; SEE IF FIRST SUBUNIT TO BE TESTED
023606                                  BEQ     3$              ; IF SO, BRANCH
023606  010000  023632                  ^O010000,3$
                                        .IF B 3$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
```

```
                                        .ENDC
023610  104647  000001                  MOV     1(R4),R0        ; GET SECOND SUBUNIT INFORMATION
023612  102207  040000                  BIT     #40000,R0       ; SEE IF THIS SUBUNIT IS USED
023614                                  BEQ     3$              ; IF SO, BRANCH
023614  010000  023632                  ^O10000,3$
                                        .IF B 3$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023616  104647  000002                  MOV     2(R4),R0        ; SEE IF THIRD SUBUNIT TO BE TESTED
023620  102207  040000                  BIT     #40000,R0       ; CHECK TESTING BIT
023622                                  BEQ     3$              ; IF IT IS TO BE TESTED, BRANCH
023622  010000  023632                  ^O10000,3$
                                        .IF B 3$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023624  104647  000003                  MOV     3(R4),R0        ; SEE IF FOURTH SUBUNIT TO BE TESTED
023626  102207  040000                  BIT     #40000,R0       ; TEST TESTING BIT
023630                                  BNE     NOU             ; IF UNIT (THE WHOLE THING) NOT TO BE TESTED, BRANCH
023630  050000  023674                  ^O50000,NOU
                                        .IF B NOU
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023632  104207  000072          3$:     MOV     #TLEN.U,R0      ; R0 HAS UNIT DATA STRUCTURE LENGTH
023634                                  CALL    GETMEM          ; GET MEMORY
023634  020000  022765                  ^O20000,GETMEM
                                        .IF B GETMEM
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023636  100707  176772                  MOV     R0,@LASTU       ; LINK LAST POINTER TO THIS ONE
023640                                  ASSUME  U.NEXT,0
                                        .IF     NE,U.NEXT-0
                                        .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                        .ENDC
023640  104070  022632                  MOV     R0,LASTU        ; SAVE POINTER TO THIS ONE
023642  100672  000025                  MOV     R2,U.MASK(R0)   ; SAVE PORT SELECT MASK
023644  104203  000004                  MOV     #4,R3           ; MAXIMUM OF FOUR SUBUNITS
023646                                  PUSH    R0              ; SAVE POINTER TO UNIT INFORMATION
                                        .IRP X,<R0>
                                                                        MOV X,-(SP)
                                        .ENDR
023646  100467                                                         MOV R0,-(SP)
023647  104145
023650  103205  170000                  MOV     (R4),R5         ; GET STARTING SUBUNIT NUMBER
023652  100675  000063                  BIC     #^CHBHINB,R5    ; CLEAR 'NOT USED' BIT, IF SET
023654  105207  000001                  MOV     R5,U.UNUM(R0)   ; SAVE STARTING SUBUNIT NUMBER
023656  104245                  1$:     ADD     #U.SUBP,R0      ; R0 WILL POINT TO SUBUNIT POINTERS
023657  102205  040000                  MOV     (R4)+,R5        ; MOVE SUBUNIT NUMBER TO UNIT PARAMETERS
023661                                  BIT     #40000,R5       ; SEE IF SUBUNIT TO BE TESTED
023661  010000  023665                  BEQ     2$              ; IF SO, BRANCH
                                        ^O10000,2$
                                        .IF B 2$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023663  104205  177777                  MOV     #-1,R5 - - -    ; FLAG SUBUNIT AS NOT TESTED
023665  100275                  2$:     MOV     R5,(R0)+        ; MOVE TO SUBUNIT POINTERS (NON-NEG IF TESTED)
023666  117403                          DEC     R3              ; DECREMENT COUNT
023667                                  BNE     1$              ; IF ALL SUBUNITS ARE FILLED IN, BRANCH
023667  050000  023656                  ^O50000,1$
```

```
                                        .IF B 1$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023671                                  POP     R0              ; RESTORE R0
                                        .IRP X,<R0>
                                                                                MOV (SP)+,X
                                        .ENDR
                                                                                MOV (SP)+,R0
023671  104267
023672  107204  000004                 SUB     #4,R4           ; R4 NOW POINTS TO BEGINNING OF UNIT HANDLED
023674  104302  022633          NOU:    MOV     UMASK,R2        ; GET UNIT PORT MASK
023676  110202                          ROL     R2              ; ROTATE TO NEXT PORT
023677  104020  022633                 MOV     R2,UMASK        ; SAVE MASK
023701  105204  000004                 ADD     #4,R4           ; R4 POINTS TO NEXT UNIT INFORMATION
023703  106204  023573                 CMP     #UNITS+16.,R4   ; SEE IF ALL UNITS SET UP
023705                                  BNE     ULOP            ; IF ALL UNITS NOT SETUP, BRANCH
023705  050000  023601                 ^O50000,ULOP
                                        .IF B ULOP
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023707  104203  037705                 MOV     #FIRSTU,R3      ; COMPLETE RING
023711  100703  176717                 MOV     R3,@LASTU       ; LAST UNIT NOW POINTS TO FIRST
023713                                  ASSUME  U.NEXT,0
                                        .IF     NE,U.NEXT-0
                                        .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                        .ENDC
023713  104207  037705                 MOV     #FIRSTU,R0      ; R0 POINTS TO FIRST UNIT PARAMETERS
023715                          SUSETL: PUSH    R0              ; SAVE POINTER TO UNIT PARAMETERS
                                        .IRP X,<R0>
                                                                                MOV X,-(SP)
                                        .ENDR
                                                                                MOV R0,-(SP)
023715  100467
                                ;
                                .SBTTL  BLDSUS - BUILD ALL SUBUNIT PARAMETER BLOCKS ON THIS UNIT
                                ;BLDSUS
                                ;
                                ;       BUILD ALL SUBUNIT PARAMETERS
                                ;
023716  104204  000001                  MOV     #U.SUBP,R4      ; R4 WILL POINT TO SUBUNIT POINTERS
023720  105074                          ADD     R0,R4           ; R4 POINTS TO SUBUNIT POINTERS
023721  104670  000063  022631          MOV     U.UNUM(R0),UCURSR       ; SAVE SUBUNIT NUMBER
023724  104205  000004                  MOV     #4,R5           ; MAXIMUM NUMBER OF SUBUNITS
023726  104147                  SLOP:   MOV     (R4),R0         ; R0 GETS SUBUNIT 'POINTER'
023727                                  BMI     NOSUN           ; IF SUBUNIT INACTIVE, BRANCH
023727  070000  024251                  ^O70000,NOSUN
                                        .IF B NOSUN
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
023731  104070  022631                  MOV     R0,UCURSR       ; UCURSR GETS SUBUNIT NUMBER
023733                                  PUSH    <R3,R4,R5>      ; SAVE R3 - R5
                                        .IRP X,<R3,R4,R5>
                                                                                MOV X,-(SP)
                                        .ENDR
023733  100463                                                                  MOV R3,-(SP)
023734  100464                                                                  MOV R4,-(SP)
023735  100465                                                                  MOV R5,-(SP)
                                .SBTTL  BLDBES - FIND HOW MANY WORDS NEEDED FOR THE BEGIN/END OR TRACK/GROUP SETS
                                ;BLDBES
```

```
                                        ;
                                        ;       COMPUTE THE BEGIN/END SET AREA NEEDED
                                        ;
023736  104300  022631  001576           MOV     UCURSR,HRQ.01   ; MOVE SUBUNIT NUMBER TO COMMUNICATION AREA
023741  104207  060004                   MOV     #T4UPRM,R0      ; MOVE REQUEST NUMBER TO R0
023743                                   CALL    HOSTRQ          ; REQUEST INFORMATION FROM HOST
023743  020000  001543                   ^O20000,HOSTRQ
                                         .IF B HOSTRQ
                                         .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                         .ENDC
023745  102200  000040  001576           BIT     #BEUSED,HRQ.01  ; SEE IF BEGIN/END SETS ARE USED
023750                                   BNE     2$              ; IF SO, BRANCH
023750  050000  023760                   ^O50000,2$
                                         .IF B 2$
                                         .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                         .ENDC
023752  104207  000020                   MOV     #S.TGSS+1,R0    ; MAXIMUM CONFIGURATION LENGTH
023754  105307  001604                   ADD     HRQ.07,R0       ; ADD NUMBER OF T/G SETS
023756                                   BR      BLDBB           ; BRANCH
023756  000000  023772                   ^O00,BLDBB
                                         .IF B BLDBB
                                         .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                         .ENDC
023760  104207  000013           2$:     MOV     #S.BESS,R0      ; MINIMUM CONFIGURATION WORD COUNT
023762  104301  001600                   MOV     HRQ.03,R1       ; GET NUMBER OF BEGIN/END SETS
023764                                   BNE     1$              ; IF NON-ZERO, BRANCH
023764  050000  023767                   ^O50000,1$
                                         .IF B 1$
                                         .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                         .ENDC
023766  115401                           INC     R1              ; MAKE R1 1
023767  105011                   1$:     ADD     R1,R1           ; MAKE B/E SETS * 2
023770  105011                           ADD     R1,R1           ; MAKE B/E SETS * 4
023771  105017                           ADD     R1,R0           ; ADD TO LENGTH OF SUBUNIT PARAMETERS
                                 .SBTTL  BLDBB  - FIND HOW MANY WORDS NEEDED FOR THE BAD BLOCKS
023772                           BLDBB:
                                        ;
                                        ;       COUNT THE BAD BLOCK PARAMETERS
                                        ;
023772                                   PUSH    R0              ; SAVE SUBUNIT LENGTH
                                         .IRP X,<R0>
                                                                                 MOV X,-(SP)
                                         .ENDR
023772  100467                                                                  MOV R0,-(SP)
023773  104300  022631  001576           MOV     UCURSR,HRQ.01   ; MOVE SUBUNIT NUMBER TO COMMUNICATION AREA
023776  104207  060005                   MOV     #T4BB1,R0       ; MOVE REQUEST NUMBER TO R0
024000                                   CALL    HOSTRQ          ; REQUEST INFORMATION FROM HOST
024000  020000  001543                   ^O20000,HOSTRQ
                                         .IF B HOSTRQ
                                         .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                         .ENDC
024002                                   POP     R0              ; RESTORE SUBUNIT LENGTH
                                         .IRP X,<R0>
                                                                                 MOV (SP)+,X
                                         .ENDR
024002  104267                                                                  MOV (SP)+,R0
024003  104301  001576                   MOV     HRQ.01,R1       ; GET NUMBER OF BAD BLOCKS
```

```
024005  104010  022634                  MOV     R1,NUMBB        ; SAVE NUMBER OF BAD BLOCKS
024007  105011                          ADD     R1,R1           ; MAKE BAD BLOCKS * 2
024010  105017                          ADD     R1,R0           ; ADD TO SUBUNIT DATA STRUCTURE LENGTH
024011                                  CALL    GETMEM          ; GET REQUESTED AMOUNT OF MEMORY
024011  020000  022765                  ^O20000,GETMEM
                                        .IF B GETMEM
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
                                .SBTTL  COPYSU - COPY ALL SUBUNIT PARAMETERS TO SUBUNIT BLOCK
                                ;COPYSU
                                ;
                                ;       COPY SUBUNIT PARAMETRS TO SUBUNIT BLOCK
                                ;
024013                                  PUSH    R0              ; SAVE R0
                                        .IRP X,<R0>
                                                                        MOV X,-(SP)
                                        .ENDR
024013  100467                                                          MOV R0,-(SP)
024014  104300  022631  001576          MOV     UCURSR,HRQ.01   ; MOVE SUBUNIT NUMBER TO COMMUNICATION AREA
024017  104207  060004                  MOV     #T4UPRM,R0      ; MOVE REQUEST NUMBER TO R0
024021                                  CALL    HOSTRQ          ; REQUEST INFORMATION FROM HOST
024021  020000  001543                  ^O20000,HOSTRQ
                                        .IF B HOSTRQ
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
024023                                  POP     R0              ; RESTORE R0
                                        .IRP X,<R0>
                                                                        MOV (SP)+,X
                                        .ENDR
024023  104267                                                          MOV (SP)+,R0
                                ;
                                ;       THESE ARE THE ONLY BITS THAT THE HOST SHOULD SEND TO THE TEST
                                ;
        077177                  VALBIT  =       INITW!DCYLS!ECCCHK!RONLY!WONLY!RTRIES!SEQSEK!BEUSED!TRACKS!WCHECK!WCHKAL!DAT
                                ;
024024  103200  100600  001576          BIC     #^CVALBIT,HRQ.01        ; CLEAR ALL BUT VALID BITS (SUBUNIT PARAMETERS)
024027  104301  001577                  MOV     HRQ.02,R1       ; GET PATTERN NUMBER
024031  100671  000004                  MOV     R1,S.PAT(R0)    ; SAVE
024033  104305  001576                  MOV     HRQ.01,R5       ; GET UNIT PARAMETERS
024035  102205  020000                  BIT     #DCYLS,R5       ; SEE IF DIAGNOSTIC CYLINDERS ARE USED
024037                                  BEQ     4$              ; IF NOT, BRANCH
024037  010000  024047                  ^O10000,4$
                                        .IF B 4$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
024041  102205  004000                  BIT     #RONLY,R5       ; SEE IF READ ONLY
024043                                  BNE     4$              ; IF SO, BRANCH
024043  050000  024047                  ^O50000,4$
                                        .IF B 4$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
024045  101205  040000                  BIS     #INITW,R5       ; FLAG UNIS AS INITIALLY WRITTEN
024047  102205  000040          4$:     BIT     #BEUSED,R5      ; SEE IF BEGIN/END SETS USED
024051                                  BNE     3$              ; IF SO, BRANCH
024051  050000  024117                  ^O50000,3$
                                        .IF B 3$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
```

```
                                        .ENDC
024053  100175                          MOV     R5,(R0)         ; SAVE IN SUBUNIT AREA
024054                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
                                        .IF     NE,S.PARM-0
                                        .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                        .ENDC
024054  104201  000013                  MOV     #S.BESS,R1      ; R1 WILL POINT TO BEGIN/END SET AREA
024056  105071                          ADD     R0,R1           ; R1 POINTS TO TRACK/GROUP AREA
024057  104202  001600                  MOV     #HRQ.03,R2      ; R2 POINTS TO START/END CYL
024061  104225                          MOV     (R2)+,R5        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
024062  100615  000002                  MOV     R5,2(R1)        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
024064  104225                          MOV     (R2)+,R5        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
024065  100615  000003                  MOV     R5,3(R1)        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
024067  104225                          MOV     (R2)+,R5        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
024070  100115                          MOV     R5,(R1)         ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
024071  104225                          MOV     (R2)+,R5        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
024072  100615  000001                  MOV     R5,1(R1)        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
024074  105201  000004                  ADD     #4,R1           ; R1 POINTS AFTER LIMITING CYLINDERS
024076  104224                          MOV     (R2)+,R4        ; GET TRACK/GROUP COUNT
024077  104225                  5$:     MOV     (R2)+,R5        ; GET WORD
024100  100215                          MOV     R5,(R1)+        ; SAVE
024101  117404                          DEC     R4              ; DECREMENT COUNT
024102                                  BNE     5$              ; IF INCOMPLETE, BRANCH
024102  050000  024077                  ^O50000,5$
                                        .IF B 5$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
024104  104415                          MOV     -(R1),R5        ; GET LAST T/G
024105  101205  100000                  BIS     #100000,R5      ; MARK AS END OF LIST
024107  100115                          MOV     R5,(R1)         ; SAVE
024110  104201  000020                  MOV     #S.TGSS+1,R1    ; R1 WILL POINT TO START OF T/G LIST
024112  105071                          ADD     R0,R1           ; R1 POINTS TO START OF T/G LIST
024113  105301  001604                  ADD     HRQ.07,R1       ; R1 NOW POINTS TO AFTER T/G'S (FOR BAD BLOCKS)
024115                                  BR      COPBB           ; BRANCH
024115  000000  024157                  ^O0,COPBB
                                        .IF B COPBB
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
024117  104201  000013          3$:     MOV     #S.BESS,R1      ; R1 WILL POINT TO BEGIN/END SETS
024121  105071                          ADD     R0,R1           ; R1 POINTS TO BEGIN/END SETS
024122  104202  001600                  MOV     #HRQ.03,R2      ; POINT TO BEGIN/END SET COUNT
024124  104224                          MOV     (R2)+,R4        ; GET COUNT, SEE IF NON-ZERO
024125                                  BNE     2$              ; IF SO, BRANCH
024125  050000  024132                  ^O50000,2$
                                        .IF B 2$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
024127  115404                          INC     R4              ; SKIP COUNT
024130  101205  000200                  BIS     #ONLYCL,R5      ; FLAG AS BEGIN/END CYLINDER SUPPLIED
024132  100175                  2$:     MOV     R5,(R0)         ; SAVE IN SUBUNIT AREA
024133                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
                                        .IF     NE,S.PARM-0
                                        .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                        .ENDC
024133  104225                  1$:     MOV     (R2)+,R5        ; GET LO STARTING CYL
024134  100615  000002                  MOV     R5,2(R1)        ; SAVE
024136  104225                          MOV     (R2)+,R5        ; GET HI STARTING CYL
```

```
024137  100615  000003                  MOV     R5,3(R1)        ; SAVE
024141  104225                          MOV     (R2)+,R5        ; GET LO ENDING CYL
024142  100115                          MOV     R5,(R1)         ; SAVE
024143  104225                          MOV     (R2)+,R5        ; GET HI ORDER ENDING CYL
024144  100615  000001                  MOV     R5,1(R1)        ; SAVE
024146  105201  000004                  ADD     #4,R1           ; R1 POINTS AFTER BEGIN/END CYLINDERS
024150  117404                          DEC     R4              ; DECREMENT COUNT
024151                                  BNE     1$              ; LOOP AND COPY
024151  050000  024133                  ^050000,1$
                                        .IF B 1$
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
024153  104415                          MOV     -(R1),R5        ; MOVE EOL TO BEGIN/END LIST
024154  101205  100000                  BIS     #100000,R5      ; SET HI BIT TO FLAG EOL
024156  100215                          MOV     R5,(R1)+        ; SAVE
024157                          LOPBB:  PUSH    R0              ; SAVE R0
                                        .IRP X,<R0>
                                                                                MOV X,-(SP)
                                        .ENDR
024157  100467                                                                 MOV R0,-(SP)
024160  104300  022631  001576          MOV     UCURSR,HRQ.01   ; MOVE TO COMMUNICATION AREA
024163  104207  060005                  MOV     #T4BB1,R0       ; MOVE REQUEST NUMBER TO R0
024165                                  CALL    HOSTRQ          ; REQUEST INFORMATION FROM HOST
024165  020000  001543                  ^020000,HOSTRQ
                                        .IF B HOSTRQ
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
024167                                  POP     R0              ; RESTORE R0
                                        .IRP X,<R0>
                                                                                MOV (SP)+,X
                                        .ENDR
024167  104267                                                                 MOV (SP)+,R0
024170  104202  001577                  MOV     #HRQ.02,R2      ; R2 POINTS AT BAD BLOCKS
024172  104304  022634                  MOV     NUMBB,R4        ; R4 IS NUMBER OF BAD BLOCKS
024174                                  BEQ     NOBB            ; IF NO BAD BLOCKS, BRANCH
024174  010000  024237                  ^010000,NOBB
                                        .IF B NOBB
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
024176  100671  000012                  MOV     R1,S.BADP(R0)   ; MOVE POINTER TO BAD BLOCKS TO SUB PARAM
024200  114004                          CLR     R4              ; START LOOP AT ZERO
024201  106204  000016          BBLOP:  CMP     #14.,R4         ; SEE IF SECOND BLOCK NEEDED
024203                                  BNE     NOEXTR          ; IF NOT, BRANCH
024203  050000  024220                  ^050000,NOEXTR
                                        .IF B NOEXTR
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
024205                                  PUSH    R0              ; SAVE R0
                                        .IRP X,<R0>
                                                                                MOV X,-(SP)
                                        .ENDR
024205  100467                                                                 MOV R0,-(SP)
024206  104300  022631  001576          MOV     UCURSR,HRQ.01   ; SAVE IN COMMUNICATIUN AREA
024211  104207  060006                  MOV     #T4BB2,R0       ; LOAD REQUEST NUMBER
024213                                  CALL    HOSTRQ          ; REQUEST FURTHER BLOCKS FROM HOST
024213  020000  001543                  ^020000,HOSTRQ
                                        .IF B HOSTRQ
```

```
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
024215                                  POP     R0              ; RESTORE R0
                                        .IRP X,<R0>
                                                                                MOV (SP)+,X
                                        .ENDR
024215  104267                                                                  MOV (SP)+,R0
024216  104202  001576                  MOV     #HRQ.01,R2      ; POINT R2 TO BAD BLOCKS
024220  104225          NOEXTR: MOV     (R2)+,R5        ; GET BAD BLOCK
024221  100215                  MOV     R5,(R1)+        ; SAVE BAD BLOCK
024222  104225                  MOV     (R2)+,R5        ; GET BAD BLOCK
024223  100215                  MOV     R5,(R1)+        ; SAVE BAD BLOCK
024224  115404                  INC     R4              ; INCREMENT R4
024225  106304  022634          CMP     NUMBB,R4        ; SEE IF ALL BLOCKS COPIED
024227                          BNE     BBLOP           ; IF NOT, BRANCH
024227  050000  024201          ^O50000,BBLOP
                                .IF B BBLOP
                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                .ENDC
024231  104415                  MOV     -(R1),R5        ; GET LAST BAD BLOCK
024232  101205  100000          BIS     #100000,R5      ; FLAG AS EOL
024234  100215                  MOV     R5,(R1)+        ; SAVE EOL
024235                          BR      CPYBEX          ; BRANCH
024235  000000  024241          ^O0,CPYBEX
                                .IF B CPYBEX
                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                .ENDC
024237  100674  000012  NOBB:   MOV     R4,S.BADP(R0)   ; FLAG AS NO BAD BLOCKS
024241  114004          CPYBEX: CLR     R4              ; CLEAR R4
024242  100674  000007          MOV     R4,S.SCHR(R0)   ; FLAG CHARACTERISTICS AS NOT YET DEFINED
                        ;
024244                          POP     <R5,R4,R3>      ; RESTORE R3 - R5
                                .IRP X,<R5,R4,R3>
                                                                                MOV (SP)+,X
                                .ENDR
024244  104265                                                                  MOV (SP)+,R5
024245  104264                                                                  MOV (SP)+,R4
024246  104263                                                                  MOV (SP)+,R3
024247                          BR      BSUSEX          ; BRANCH
024247  000000  024253          ^O0,BSUSEX
                                .IF B BSUSEX
                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                .ENDC
024251  104207  177777  NOSUN:  MOV     #-1,R0          ; TO FLAG SUBUNIT AS INACTIVE
024253  100247          BSUSEX: MOV     R0,(R4)+        ; MOVE POINTER TO SUB PARAM TO UNIT PARAMS
024254  115400  022631          INC     UCURSR          ; MOVE TO NEXT SUBUNIT
024256  117405                  DEC     R5              ; DECREMENT COUNT
024257                          BNE     SLOP            ; IF UNEXPIRED, BRANCH
024257  050000  023726          ^O50000,SLOP
                                .IF B SLOP
                                .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                .ENDC
024261                          POP     R0              ; RESTORE UNIT POINTER
                                .IRP X,<R0>
                                                                                MOV (SP)+,X
                                .ENDR
024261  104267                                                                  MOV (SP)+,R0
```

```
024262  104177                          MOV     (R0),R0         ; GO TO NEXT UNIT
024263                                  ASSUME  U.NEXT,0
                                        .IF     NE,U.NEXT-0
                                        .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                        .ENDC
024263  103207  140000                  BIC     #UNADDR,R0      ; CLEAR UNUSED ADDRESSING BITS
024265  106207  037705                  CMP     #FIRSTU,R0      ; SEE IF THE ENTIRE RING IS SET UP
024267                                  BNE     SUSETL          ; IF NOT, BRANCH
024267  050000  023715                  ^050000,SUSETL
                                        .IF B SUSETL
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
024271                                  RETURN
024271                                  JMP     0
024271  000000  000000                  ^00,0
                                        .IF B 0
                                        .ERROR  ; BRANCH/JUMP ADDRESS MUST BE DEFINED
                                        .ENDC
    109 024273                          TEXT
                                        .LIST   ME
                                .SBTTL  ********** NON-LOADED OVERLAY, ERROR MESSAGES
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;************* *****************************************************************
                                ;
                                ;       ERROR MESSAGE FORMAT BUFFERS (NEVER LOADED INTO UDA)
                                ;
024273                                  DMOVLY  MS,0                    ; ERROR MESSAGE OVERLAY
024273                                  OVTERM          ;TERMINATE LAST OVERLAY
024273  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
        022710                  OVL.MN  =       .-OVE.MN
        022710                  OVL...  =       OVL...+OVL.MN
        023330                  OV...   =       OV...+OVL.MN
                                        .CLEDC          ;CLEAR EDC FOR NEXT OVERLAY
                                .MACRO  OVTERM
                                        .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                OVL.MS  =       .-OVE.MS
                                OVL...  =       OVL...+OVL.MS
                                OV...   =       OV...+OVL.MS
                                .ENDM
        046660                  OVS.MS  =       OV...*2
        000000                  OVE.MS  =       0
                                        .OFSET  OV...-0
                                        .NLIST  BEX
                                        .LIST   MEB
000000      042     101     124 ER1:    .ASCII  \"ATTN ASSERTED DURING SEEK'N\
000016      042     123     105         .ASCII  \"SEEK FROM GRP 'D8" CYL 'D28" TO GRP 'D8" CYL 'D28NR1\
000050      000                         .BYTE   0
000051      042     122     105 SER22:  .ASCII  \"REAL TIME STATE 'H16N\
000064      042     123     124         .ASCII  \"STATUS (R TO L): 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
000116      000                         .BYTE   0
000117      042     101     124 ER2:    .ASCII  \"ATTN ASSERTED UNEXPECTEDLY, ASYNC DRIVE ERROR OR LOGGABLE'N\
000155      042     111     116         .ASCII  \"INFORMATION'NR1\
000165      000                         .BYTE   0
000166      042     123     105 ER3:    .ASCII  \"SEEK DID NOT COMPLETE, NEITHER ATTN OR R/W RDY WAS ASSERTED'N\
000225      042     102     105         .ASCII  \"BEFORE TIMEOUT'N\
```

```
000235  042  123  105          .ASCII  \"SEEK FROM GRP 'D8" CYL 'D28" TO GRP 'D8" CYL 'D28NR1\
000270  000                     .BYTE   0
000271  042  122  103  ER4:     .ASCII  \"RCT AREA CORRUPTED, COULD NOT FIND REPLACEMENT FOR'N\
000323  042  114  102          .ASCII  \"LBN "R1\
000327  042  101  124          .ASCII  \"ATTEMPTED TO READ RCT LBN 'D28N\
000347  042  123  105          .ASCII  \"SEARCING FOR LBN"S10D28N\
000364  000                     .BYTE   0
000365  042  110  105  ER5:     .ASCII  \"HEADER NOT FOUND DURING WRITE'N\
000405  122  061  042          .ASCII  \R1"BN "D28NR3\
000413  042  123  105          .ASCII  \"SECTORS FROM INDEX 'D8" TRK 'D8" GRP 'D8" CYL 'D28N\
000445  042  117  122          .ASCII  \"ORIGIN OF SEEK: GRP 'D8" CYL 'D28N\
000467  000                     .BYTE   0
000470  042  101  124  SER19:  .ASCII  \"ATTEMPT 'D3N\
000476  122  061  042          .ASCII  \R1"BN "D28NR3\
000505  042  123  105          .ASCII  \"SECTORS FROM INDEX 'D8" TRK 'D8" GRP 'D8" CYL 'D28N\
000537  042  117  122          .ASCII  \"ORIGIN OF SEEK: GRP 'D8" CYL 'D28NR1\
000561  000                     .BYTE   0
000562  042  123  105  ER6:     .ASCII  \"SELECT TRACK AND WRITE LEVEL 1 CMD NOT EXECUTED'NR1\
000614  000                     .BYTE   0
000615  042  105  103  ER7:     .ASCII  \"ECC DETECTED ERROR'NR1\
000630  000                     .BYTE   0
000631  042  105  103  ER8:     .ASCII  \"ECC DETECTED ERROR, BUT CORRECTION FAILED'NR1\
000660  000                     .BYTE   0
000661  042  122  105  SER21:  .ASCII  \"RETRY 'D4N\
000666  042  105  122          .ASCII  \"ERROR RECOVERY LEVEL 'D8N\
000703  122  061  042          .ASCII  \R1"BN "D28NR3\
000712  042  123  105          .ASCII  \"SECTORS FROM INDEX 'D8" TRK 'D8" GRP 'D8" CYL 'D28N\
000744  000                     .BYTE   0
000745  042  105  103  ER9:     .ASCII  \"ECC CORRECTIONS EXCEED THRESHOLD'NR1\
000767  000                     .BYTE   0
000770  042  105  103  ER10:    .ASCII  \"ECC CORRECTION SUCCEEDED, BUT EDC DETECTS ERROR'NR1\
001022  042  105  104          .ASCII  \"EDC COMPUTED 'D16N\
001033  042  105  104          .ASCII  \"EDC READ"S5016N\
001043  000                     .BYTE   0
001044  042  122  105  ER11:    .ASCII  \"READ DID NOT SUCCEED ON ANY RLCOVERY LEVEL'N\
001072  122  061  042          .ASCII  \R1"BN "D28NR3\
001101  042  107  122          .ASCII  \"GRP 'D8" CYL 'D28N\
001112  000                     .BYTE   0
001113  042  104  101  ER12:    .ASCII  \"DATA COMPARISON FAILED'NR1\
001130  122  061  042          .ASCII  \R1"BN "D28NR3\
001137  042  123  105          .ASCII  \"SECTORS FROM INDEX 'D8" TRK 'D8" GRP 'D8" CYL 'D28N\
001171  042  120  101          .ASCII  \"PATTERN NUMBER  'D4N\
001203  042  117  106          .ASCII  \"OFFSET OF ERROR WITHIN BUFFER: 'D8N\
001225  042  117  106          .ASCII  \"OFFSET OF ERROR WITHIN DISPLAYED LIST: 'D8" (1ST WORD OFFSET 0)'N\
001266  123  064  117          .ASCII  \S4016S4016S4016S4016S4016S4016N\
001306  123  064  117          .ASCII  \S4016S4016S4016S4016S4016S4016N\
001325  000                     .BYTE   0
001326  042  105  103  SER24:  .ASCII  \"ECC OR EDC HAD DETECTED ERROR IN BUFFER'N\
001353  000                     .BYTE   0
001354  042  105  103  SER25:  .ASCII  \"ECC OR EDC HAD <<NOT>> DETECTED ERROR IN BUFFER'N\
001405  000                     .BYTE   0
001406  042  104  122  ER13:    .ASCII  \"DRIVE NOT ONLINE TO UDA, AND NOT SPINABLE'N\
001434  000                     .BYTE   0
001435  042  125  116  ER14:    .ASCII  \"UNABLE TO COMPLETE SEEK -- TRIED 3 TIMES'N\
001462  122  061  042          .ASCII  \R1"BN "D28NR3\
001471  042  107  122          .ASCII  \"GRP 'D8" CYL 'D28N\
001502  000                     .BYTE   0
```

```
001503  042  123  105  ER15:   .ASCII  \"SEEK REQUIRED '"D2" RETRIES BEFORE COMPLETING'"N\
001532  122  061  042           .ASCII  \R1"BN '"D28NR3\
001541  042  107  122           .ASCII  \"GRP '"D8" CYL '"D28N\
001552  000                     .BYTE   0
001553  042  105  122  ER16:   .ASCII  \"ERRORS DURING DRIVE INITIALIZATION AND SETUP'"NR1\
001603  000                     .BYTE   0
001604  042  116  117  ER17:   .ASCII  \"NO VALID STATE FROM DRIVE'"N\
001622  042  116  117           .ASCII  \"NO DRIVE CLOCKS'"N\
001633  000                     .BYTE   0
001634  042  116  117  ER17A:  .ASCII  \"NO VALID STATE FROM DRIVE'"N\
001652  042  110  101           .ASCII  \"HARD PARITY OR PULSE ERROR FOR 1/2 A SECOND'"N\
001701  000                     .BYTE   0
001702  042  101  124  ER18:   .ASCII  \"ATTEMPT TO WRITE ON WRITE PROTECTED DRIVE'"N\
001730  042  105  122           .ASCII  \"ERROR CODE RETURNED FROM UDA: '"O16NR1\
001753  000                     .BYTE   0
001754  042  110  105  ER19:   .ASCII  \"HEADER NOT FOUND DURING READ'"N\
001773  122  061  042           .ASCII  \R1"BN '"D28NR3\
002002  042  123  105           .ASCII  \"SECTORS FROM INDEX '"D8" TRK '"D8" GRP '"D8" CYL '"D28N\
002034  042  117  122           .ASCII  \"ORIGIN OF SEEK: GRP '"D8" CYL '"D28N\
002055  000                     .BYTE   0
002056  042  123  105  ER20:   .ASCII  \"SELECT TRACK AND READ LEVEL 1 CMD NOT SENT'"NR1\
002105  000                     .BYTE   0
002106  042  104  122  ER21:   .ASCII  \"DRIVE NOT FORMATTED IN 512 BYTE MODE -- UNABLE TO TEST'"N\
002142  042  130  102           .ASCII  \"XBN 0 MODE WORD: '"O16N\
002156  000                     .BYTE   0
002157  042  125  116  ER23:   .ASCII  \"UNABLE TO CONTINUE TESTING'"NR1R1\
002177  000                     .BYTE   0
002200  042  122  125  SER42:  .ASCII  \"RUN/STOP SWITCH OUT'"N\
002213  000                     .BYTE   0
002214  042  123  120  SER43:  .ASCII  \"SPINDLE DROPPED READY'"N\
002230  000                     .BYTE   0
002231  042  120  117  SER44:  .ASCII  \"PORT SWITCH OUT'"N\
002242  000                     .BYTE   0
002243  042  120  101  SER45:  .ASCII  \"PARITY ERRORS FOR MORE THAN A HALF SECOND'"N\
002271  000                     .BYTE   0
002272  042  105  104  ER24:   .ASCII  \"EDC DETECTED ERROR BUT ECC DID NOT'"NR1\
002315  042  105  104           .ASCII  \"EDC COMPUTED '"O16" EDC READ '"O16N\
002336  000                     .BYTE   0
002337  042  127  122  ER25:   .ASCII  \"WRITE ATTEMPTED MAXIMUM TIMES'"N\
002357  122  061  042           .ASCII  \R1"BN '"D28NR3\
002365  000                     .BYTE   0
002366  042  122  105  ER26:   .ASCII  \"READ ATTEMPTED MAXIMUM TIMES'"N\
002405  122  061  042           .ASCII  \R1"BN '"D28NR3\
002414  000                     .BYTE   0
002415  042  102  117  ER28:   .ASCII  \"BOTH READ ONLY <AND> WRITE ONLY BITS SET -- HOST ERROR'"N\
002451  000                     .BYTE   0
002452  042  125  116  ER33:   .ASCII  \"UNABLE TO CORRECTLY READ OVERLAY '"O16NR1\
002476  000                     .BYTE   0
002477  042  124  110  SER39:  .ASCII  \"THIS UDA AND ALL DRIVES ATTACHED WILL BE REMOVED FROM TESTING'"N\
002537  000                     .BYTE   0
002540  042  123  105  ER34:   .ASCII  \"SERDES OVERRUN ERROR DURING READ'"NR1\
002562  000                     .BYTE   0
002563  042  104  101  ER35:   .ASCII  \"DATA OR STATE CLOCK TIMEOUT DURING READ'"NR1\
002611  000                     .BYTE   0
002612  042  104  101  ER36:   .ASCII  \"DATA SYNC TIMEOUT DURING READ'"NR1\
002633  000                     .BYTE   0
002634  042  122  057  ER37:   .ASCII  \"R/W RDY DROPPED BEFORE/DURING READ'"NR1\
```

```
002657  000                             .BYTE   0
002660  042  122  103  ER38:           .ASCII  \"RCVR RDY DROPPED BEFORE/DURING READ"NR1\
002704  000                             .BYTE   0
002705  042  101  114  ER40:           .ASCII  \"ALL COPIES OF RCT READ WITH ERROR, SEARCHING FOR"N\
002736  042  114  102                   .ASCII  \"LBN "R1\
002742  042  114  101                   .ASCII  \"LAST RCT LBN SEARCHED "D28N\
002760  042  123  105                   .ASCII  \"SEARCING FOR LBN"S6D28N\
002774  000                             .BYTE   0
002775  042  103  117  ER41:           .ASCII  \"COULD NOT FIND REPLACEMENT FOR"N\
003015  042  114  102                   .ASCII  \"LBN "R1\
003021  042  114  102                   .ASCII  \"LBN TO REPLACE "D28N\
003034  000                             .BYTE   0
003035  042  124  110  SER26:          .ASCII  \"THAT WAS REVECTORED"N\
003050  000                             .BYTE   0
003051  042  127  111  SER32:          .ASCII  \"WITH HEADER NOT FOUND"N\
003065  000                             .BYTE   0
003066  042  124  111  ER42:           .ASCII  \"TIMEOUT WAITING FOR SECTOR OR INDEX PULSE"N\
003114  042  107  122                   .ASCII  \"GRP "D8" CYL "D28NR1\
003126  000                             .BYTE   0
003127  042  123  105  ER44:           .ASCII  \"SEEK OR HEAD SELECT ERROR DETECTED DURING WRITE"NR1\
003161  000                             .BYTE   0
003162  042  123  105  ER45:           .ASCII  \"SEEK OR HEAD SELECT ERROR DETECTED DURING READ"NR1\
003213  000                             .BYTE   0
003214  042  104  101  ER47:           .ASCII  \"DATA OR STATE CLOCK TIMEOUT DURING WRITE"NR1\
003242  000                             .BYTE   0
003243  042  122  057  ER48:           .ASCII  \"R/W RDY DROPPED BEFORE/DURING WRITE"NR1\
003267  000                             .BYTE   0
003270  042  122  103  ER49:           .ASCII  \"RCVR RDY DROPPED BEFORE/DURING WRITE"NR1\
003314  000                             .BYTE   0
003315  122  061  042  ER50:           .ASCII  \R1"BEGIN/END SET STARTING BLOCK NUMBER GREATER THAN ENDING BLOCK NUMBER"N\
003361  000                             .BYTE   0
003362  122  061  042  ER51:           .ASCII  \R1"THE BEGIN/END SETS GIVEN OVERLAP"N\
003404  000                             .BYTE   0
003405  122  061  042  ER52:           .ASCII  \R1"BEGIN/END SET ENDING BLOCK NUMBER EXCEEDS MAXIMUM"N\
003440  042  115  101                   .ASCII  \"MAXIMUM BLOCK NUMBER ON DEVICE IS "D28N\
003464  000                             .BYTE   0
003465  122  061  042  ER53:           .ASCII  \R1"DUPLICATE BAD BLOCKS"N\
003501  000                             .BYTE   0
003502  122  061  042  ER54:           .ASCII  \R1"BAD BLOCK NUMBER EXCEEDS MAXIMUM. MAXIMUM BLOCK NUMBER"N\
003537  042  117  116                   .ASCII  \"ON DEVICE IS "D28N\
003551  000                             .BYTE   0
003552  122  061  042  ER55:           .ASCII  \R1"STARTING CYLINDER GREATER THAN ENDING CYLINDER"N\
003603  000                             .BYTE   0
003604  122  061  042  ER56:           .ASCII  \R1"RANDOM AND SEQUENTIAL SEEKS CAN NOT BE MIXED WITHIN A UNIT"N\
003643  000                             .BYTE   0
003644  122  061  042  ER57:           .ASCII  \R1"OVERFLOW WHEN CALCULATING THE L/DBN FROM THE GIVEN CYLINDER"N\
003704  042  103  131                   .ASCII  \"CYLINDER TOO LARGE"N\
003716  000                             .BYTE   0
003717  122  061  122  ER58:           .ASCII  \R1R1  EXCEEDS MAXIMUM FOR DEVICE. MAXIMUM IS "D8N\
003747  000                             .BYTE   0
003750  122  061  042  ER59:           .ASCII  \R1"TWO IDENTICAL "R1"S"N\
003764  000                             .BYTE   0
003765  122  061  122  ER62:           .ASCII  \R1R1"BN COMPUTED FROM END CYLINDER GIVEN EXCEEDS MAXIMUM "R1"BN ON"N\
004027  042  104  105                   .ASCII  \"DEVICE - CYLINDER TOO LARGE"N\
004046  000                             .BYTE   0
004047  042  117  120  SER20:          .ASCII  \"OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT"N\
004114  000                             .BYTE   0
```

```
004115  042  122  105  ER63:    .ASCII  \"REAL TIME STATE RECEIVE ERROR DURING WRITE"NR1\
004144  000                     .BYTE   0
004145  042  122  105  ER64:    .ASCII  \"REAL TIME STATE RECEIVE ERROR DURING READ"NR1\
004174  000                     .BYTE   0
004175  042  125  116  ER68:    .ASCII  \"UNKNOWN ERROR CODE DURING WRITE"NR1\
004217  000                     .BYTE   0
004220  042  125  116  ER69:    .ASCII  \"UNKNOWN ERROR CODE DURING READ"NR1\
004241  000                     .BYTE   0
004242  042  105  122  SER36:   .ASCII  \"ERROR CODE RETURNED "D16NR1\
004260  000                     .BYTE   0
004261  042  124  111  ER70:    .ASCII  \"TIMEOUT OF SEND"NR1R1\
004274  000                     .BYTE   0
004275  042  124  111  ER71:    .ASCII  \"TIMEOUT OF RECEIVE"NR1R1\
004311  000                     .BYTE   0
004312  042  106  111  ER72:    .ASCII  \"FIRST WORD RECEIVED WAS NOT START FRAME"NR1R1\
004341  000                     .BYTE   0
004342  042  106  122  ER73:    .ASCII  \"FRAMING ERROR ON LEVEL 0 RECEIVE"NR1R1\
004365  000                     .BYTE   0
004366  042  103  110  ER74:    .ASCII  \"CHECKSUM ERROR ON LEVEL 0 RECEIVE"NR1R1\
004412  000                     .BYTE   0
004413  042  102  125  ER75:    .ASCII  \"BUFFER SIZE SMALLER THAN LEVEL 2 RESPONSE"NR1R1\
004443  000                     .BYTE   0
004444  042  122  105  ER76:    .ASCII  \"RESPONSE OF LEVEL 2 CMD NOT AS EXPECTED"NR1\
004472  042  105  130            .ASCII  \"EXPECTED RESPONSE "H8N\
004505  042  122  105            .ASCII  \"RESPONSE RECEIVED "H8NR1\
004522  000                     .BYTE   0
004523  042  104  122  ER77:    .ASCII  \"DRIVE NEVER DEASSERTED RECEIVER READY AFTER SEND"NR1R1\
004556  000                     .BYTE   0
004557  042  125  116  ER78:    .ASCII  \"UNKNOWN ERROR CODE RETURNED FROM LEVEL 2 RECEIVE"NR1\
004611  042  105  122            .ASCII  \"ERROR CODE RETURNED "D16NR1\
004627  000                     .BYTE   0
004630  042  101  124  SER0:    .ASCII  \"ATTEMPTING TO GET STATUS"N\
004645  000                     .BYTE   0
004646  042  101  124  SER1:    .ASCII  \"ATTEMPTING DRIVE CLEAR CMD"N\
004664  000                     .BYTE   0
004665  042  101  124  SER2:    .ASCII  \"ATTEMPTING TO BRING DRIVE ONLINE"N\
004706  000                     .BYTE   0
004707  042  101  124  SER3:    .ASCII  \"ATTEMPTING TO CHANGE MODE"N\
004725  000                     .BYTE   0
004726  042  101  124  SER4:    .ASCII  \"ATTEMPTING ERROR RECOVERY CMD"N\
004746  000                     .BYTE   0
004747  042  101  124  SER5:    .ASCII  \"ATTEMPTING TO GET SUBUNIT CHAR"N\
004767  000                     .BYTE   0
004770  042  101  124  SER6:    .ASCII  \"ATTEMPTING TO SPIN UP DRIVE"N\
005007  000                     .BYTE   0
005010  042  101  124  SER7:    .ASCII  \"ATTEMPTING TO RECALIBRATE"N\
005026  000                     .BYTE   0
005027  042  101  124  SER8:    .ASCII  \"ATTEMPTING TO GET COMMON CHAR"N\
005047  000                     .BYTE   0
005050  042  101  124  SER9:    .ASCII  \"ATTEMPTING TO ISSUE SEEK"N\
005065  000                     .BYTE   0
005066  042  125  116  ER1000:  .ASCII  \"UNABLE TO FIND REQUESTED DRIVE FOR TESTING"N\
005114  042  124  110            .ASCII  \"THE FOLLOWING IS VISIBLE ON THE PORTS"N\
005140  042  125  104            .ASCII  \"UDA PORT 0 -- "R1\
005151  042  125  104            .ASCII  \"UDA PORT 1 -- "R1\
005162  042  125  104            .ASCII  \"UDA PORT 2 -- "R1\
005173  042  125  104            .ASCII  \"UDA PORT 3 -- "R1\
```

```
005204  000                             .BYTE   0
005205  042  116  117  SER10:  .ASCII  \'NO DRIVE ATTACHED'N\
005217  000                             .BYTE   0
005220  042  122  103  SER11:  .ASCII  \'RCVR RDY NEVER ASSERTED'N\
005235  000                             .BYTE   0
005236  042  124  111  SER12:  .ASCII  \'TIMEOUT OF SEND'N\
005247  000                             .BYTE   0
005250  042  124  111  SER13:  .ASCII  \'TIMEOUT OF RECEIVE'N\
005262  000                             .BYTE   0
005263  042  106  111  SER14:  .ASCII  \'FIRST WORD RECEIVED WAS NOT START FRAME'N\
005310  000                             .BYTE   0
005311  042  106  122  SER15:  .ASCII  \'FRAMING ERROR ON LEVEL 0 RECEIVE'N\
005332  000                             .BYTE   0
005333  042  103  110  SER16:  .ASCII  \'CHECKSUM ERROR ON LEVEL 0 RECEIVE'N\
005355  000                             .BYTE   0
005356  042  122  105  SER17:  .ASCII  \'RESPONSE LONGER THAN EXPECTED FOR GET STATUS CMD'N\
005407  000                             .BYTE   0
005410  042  104  122  SER18:  .ASCII  \'DRIVE 'R1\
005415  000                             .BYTE   0
005416  104  061  062  SER18D: .ASCII  \D12'',  ''\       ; NOTE: THE STRING WITHIN THE \\'S <<MUST>>
005422  104  061  062  SER18C: .ASCII  \D12'',  ''\       ; BE AN EVEN NUMBER OF CHAR, BECAUSE THE
005426  104  061  062  SER18B: .ASCII  \D12'',  ''\       ; LABELS WILL FORCE WORD ALINGMENT, LEAVING
005432  104  061  062  SER18A: .ASCII  \D12N\            ; JUNK IN THE LAST BYTE. (SER18A OK TO BE ODD)
005434  000                             .BYTE   0
005435  042  104  122  SER40:  .ASCII  \'DRIVE NOT AVAILABLE TO THIS UDA'N\
005456  000                             .BYTE   0
005457  042  125  116  SER41:  .ASCII  \'UNSPINABLE DRIVE 'R1\
005471  000                             .BYTE   0
005472  042  122  105  RBNTXT: .ASCII  \'REVECTORED TO RBN 'D28N\
005506  000                             .BYTE   0
005507  042  124  122  TRK$:   .ASCII  \''TRACK''\
005512  000                             .BYTE   0
005513  042  107  122  GRP$:   .ASCII  \''GROUP''\
005516  000                             .BYTE   0
005517  042  114  042  L$:     .ASCII  \''L''\
005520  000                             .BYTE   0
005521  042  104  042  D$:     .ASCII  \''D''\
005522  000                             .BYTE   0
005523  042  122  103  RCTL$:  .ASCII  \''RCT L''\
005526  000                             .BYTE   0
                        .SBTTL  INFORMATIONAL MESSGES
                        ;
                        ;       MESSAGES
                        ;
005527  116  042  111  MS1:    .ASCII  \N''INITIAL WRITE COMPLETE'N\
005544  000                             .BYTE   0
005545  116  042  124  MS2:    .ASCII  \N''THE PREVIOUS DEVICE FATAL WILL CAUSE THE FOLLOWING DRIVES'N\
005603  042  124  117                  .ASCII  \''TO BE DROPPED: 'R1\
005615  000                             .BYTE   0
005616  116  042  101  MS3:    .ASCII  \N''A CORRECTABLE ECC ERROR EXISTS IN 'R1''BN 'D28N\
005646  042  123  105                  .ASCII  \''SECTORS FROM INDEX 'D8'' TRK 'D8'' GRP 'D8'' CYL 'D28N\
005700  000                             .BYTE   0
005701  116  042  122  MS4:    .ASCII  \N''READ ONLY DRIVE, INITIAL WRITE WILL NOT BE PERFORMED'N\
005735  000                             .BYTE   0
113 005735                              DMEND
    005736 000105                       .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
114        000001                       .END
```

SYMBOL TABLE

```
AFTOP   005403   CPYBEX  024241   ECCCHK= 010000   ER49    003270   GORCLB  005076
AFTWRT  010744   CR.CLR  002364   ECCFLG= 010000   ER5     000365   GORTRY  005056
ALLOCM  010271   CR.DIS  002410   ECCRSH= 000002   ER50    003315   GOSEEK  005066
ATTN  = 000002   CR.ERR  002403   ECHO  = 000010   ER51    003362   GOTOBE  005711
AVAIL = 000100   CR.GCR  022606   ECHOC = 000350   ER52    003405   GOTOFS  005714
BBLOP   024201   CR.GST  002357   EDCLOP  001647   ER53    003465   GO4IT   003722
BEEXT   006323   CR.INR  002415   ENABLE  005036   ER54    003502   GROUP   002766
BESDWN  006167   CR.MOD  002371   EOC   = 100000   ER55    003552   GRPCYL= 000002
BESUP   006321   CR.ONL  002352   ERCOV   014104   ER56    003604   GRPOFF= 000011
BEUSED= 000040   CR.RUN  022620   ERECOV= 000006   ER57    003644   GRP$    005513
BF.DAT= 000000   CR.SCR  022613   EREXT   014202   ER58    003717   GST     002430
BF.ECC= 000401   CR.SEK  002376   ERHARD= 100000   ER59    003750   HBHINB= 007777
BF.EDC= 000400   CSCEXT  014512   ERLEV = 000002   ER6     000562   HBLONB= 170377
BLDBB   023772   CURBN   002760   ERMASK= 100000   ER62    003765   HDRPRE= 000005
BLDUNT  023573   CVT   = 000020   ERMODE  003011   ER63    004115   HD.BAD= 110000
BLKCHK  002465   CYL     002764   ERR     002436   ER64    004145   HD.DBN= 140000
BREAK = 000000   CYLBN   020241   ERRLEV  002437   ER68    004175   HD.LBN= 000000
BSUSEX  024253   CYLSCR  006733   ERRMC = 060014   ER69    004220   HD.PRV= 050000
BUFARA= 017005   C2DFTL= 100000   ERRMES= 060013   ER7     000615   HD.RBN= 060000
BUFFLG= 040000   C2HARD= 040000   ERRPOS  003013   ER70    004261   HD.REV= 030000
BUFSIZ= 000043   DATCMP= 000002   ERSOFT= 140000   ER71    004275   HD.XBN= 120000
BUILDP  007746   DATPRE= 000005   ER1     000000   ER72    004312   HIBN    002755
BULDUM  002552   DBNCYL= 000022   ER10    000770   ER73    004342   HIBYTE= 177400
CALC    002604   DCEXT   005541   ER1000  005066   ER74    004366   HICYL = 000001
CALCSC  002756   DCLOCK= 000004   ER11    001044   ER75    004413   HIDBN = 000003
CBB2    022773   DCLR    002427   ER12    001113   ER76    004444   HILBN = 000002
CHAINS  003015   DCMPAL= 000001   ER13    001406   ER77    004523   HIMEM = 037777
CHGMOD= 000201   DCMPYS  005535   ER14    001435   ER78    004557   HIRBN = 000003
CHKBB   021155   DCREAD  005524   ER15    001503   ER8     000631   HISEED  003010
CHKBES  020324   DCYLS = 020000   ER16    001553   ER9     000745   HIXBN = 000002
CHKCYL  020122   DEBUG = 000000   ER17    001604   EXIT  = 000021   HOSTRQ  001543
CHKDN   004721   DIE   = 000002   ER17A   001634   FB.DAT= 000000   HRDREV= 000003
CHKECC  013437   DINIT = 000011   ER18    001702   FB.EDC= 000400   HRQ.RQ  001575
CHKEDC  013016   DIREC = 010000   ER19    001754   FCTSIZ= 000010   HRQ.01  001576
CHKTG   021346   DIS     002422   ER2     000117   FIRSTU= 037705   HRQ.02  001577
CHKUP   004756   DISCON= 000204   ER20    002056   FIXPAT  005455   HRQ.03  001600
CHNMAX  003017   DONE  = 060016   ER21    002106   FNDBES  006343   HRQ.04  001601
CHRRES= 000170   DOWNG   007525   ER23    002157   FNDLOP  006353   HRQ.05  001602
CLCLBN  005773   DOWNT   007333   ER24    002272   FNDRER  012070   HRQ.06  001603
CLCMAX  020711   DRC     002426   ER25    002337   FORMAT= 000001   HRQ.07  001604
CMPDAT  014403   DRINIT= 040000   ER26    002366   FSTOP = 100000   HRQ.08  001605
CMPEDC  001640   DROP  = 100000   ER28    002415   FTIME = 001000   HRQ.09  001606
CMPERR  014520   DRPALL  016727   ER3     000166   FTLDEV= 040000   HRQ.10  001607
CMP2    002536   DRTYPE= 000007   ER33    002452   FTLSYS= 000000   HRQ.11  001610
CMXEX   014721   DRVCLR= 000005   ER34    002540   FT.BUF= 000000   HRQ.12  001611
CNTLOP  005654   DRVEXT  022604   ER35    002563   FT.HI = 000001   HRQ.13  001612
COMCHR  017027   DRVID = 000004   ER36    002612   FT.LOW= 000002   HRQ.14  001613
COMPAR= 000006   DRVONL= 000213   ER37    002634   FWRD    014517   HRQ.15  001614
COMPDP  021664   DRVRUN= 000014   ER38    002660   GCR     022626   HRQ.16  001615
COMPLT= 000176   DSABLE  005046   ER4     000271   GETCHR= 000207   HRQ.17  001616
COMPSC  020657   DSERNM  022651   ER40    002705   GETMEM  022765   HRQ.18  001617
COPBB   024157   DUMSDI  002772   ER41    002775   GETSER  021707   HRQ.19  001620
COPFIN  010472   DUPPKT= 060000   ER42    003066   GETSTA= 000011   HRQ.20  001621
COPLP0  010445   D$      005521   ER44    003127   GETSUB= 000210   HRQ.21  001622
COPLP1  010453   D.LIMT= 000001   ER45    003162   GETU    023101   HRQ.22  001623
COPLP2  010466   D.SCHR= 000002   ER47    003214   GMPARM  023056   HRQ.23  001624
COUNTD  005664   ECC   = 000015   ER48    003243   GOINIT  005106   HRQ.24  001625
```

SYMBOL TABLE

```
HRQ.25  001626          MAXSND= 001750          PATPTR  003021          RTDSL   001453          SER16   005333
HRQ.26  001627          MEDTYP= 000006          PAT0    003041          RTRIES= 001000          SER17   005356
HRQ.27  001630          MEMPCL  003012          PAT1    003063          RUN     022625          SER18   005410
HRQ.28  001631          MESSAG= 060015          PAT10   003212          RVFAIL  015603          SER18A  005432
HRQ.29  001632          MICREV= 000003          PAT11   003234          RWRDY = 100000          SER18B  005426
HRQ.30  001633          MKLOOP  023014          PAT12   003237          RW.ANG= 000007          SER18C  005422
HRQ.31  001634          MOD     002424          PAT13   003261          RW.BUF= 000002          SER18D  005416
HRQ.32  001635          MOD512= 126736          PAT14   003303          RW.CMD= 000005          SER18E  023547
HRQ.33  001636          MOD576= 074161          PAT15   003310          RW.CPT= 000000          SER18F  017001
HRQ.34  001637          MOVDBN  002705          PAT2    003066          RW.HI = 000004          SER19   000470
ID      006707          MOVOUT  002746          PAT3    003071          RW.LOW= 000003          SER2    004665
INCODT= 000000          MOV2    006527          PAT4    003074          RW.SDI= 000006          SER20   004047
INDEX   002771          MRD   = 000016          PAT5    003116          RW.STA= 000001          SER21   000661
INITD   022427          MS1     005527          PAT6    003140          SBCRES= 000167          SER22   000051
INITW = 040000          MS2     005545          PAT7    003162          SCHAR0  017514          SER24   001326
INR     002431          MS3     005616          PAT8    003165          SCHAR1  020761          SER25   001354
INS     002432          MS4     005701          PAT9    003207          SCHRCP  020072          SER26   003035
INSEEK= 000012          MWR   = 000017          PHYSA = 001000          SCR     022627          SER3    004707
INSET   017005          M.PARM  003006          PNUM    003004          SCR1    003002          SER32   003051
INTEDC= 000105          NEWDNT  007622          PREVBE  006414          SCR2    003003          SER36   004242
INTINP= 000001          NEWEXT  014400          QDA   = 000000          SCTWRD= 000377          SER39   002477
IRECLB= 000216          NEWLEV  014205          RANDOM  005563          SDIVER= 00000C          SER4    004726
ISUEXT  016216          NEWOP   005344          RBLOCK  011754          SDI$  = 000003          SER40   005435
ISUSEK  016133          NEWSUB= 004000          RBNBN = 000200          SEARCH  015255          SER41   005457
IWIPRG= 100000          NEWUPT  007573          RBNFLG  002763          SECCHK  012134          SER42   002200
JMPRET  003746          NEXTBE  006460          RBNTRK= 000004          SECCYL  022641          SER43   002214
LASTU   022632          NLBEXT  006171          RBNTXT  005472          SECGRP  022640          SER44   002231
LBHINB= 177417          NLEV    014171          RBUFLN= 000415          SECMAX  003016          SER45   002243
LBLONB= 177760          NOBB    024237          RCBREQ= 002000          SECPTR  010300          SER5    004747
LBNCYL= 000000          NOBORO  005727          RCONT = 000000          SECTOR  002770          SER6    004770
LBNHST= 000012          NOEXTR  024220          RCTCPS= 000001          SECTRK  022637          SER7    005010
LBNTRK= 000011          NOSEEK  016047          RCTCSZ= 000014          SEC512= 000400          SER8    005027
LEVNZR  014334          NOSUB   004501          RCTL$   005523          SEC576= 000440          SER9    005050
LEVUSD= 020000          NOSUN   024251          RCV   = 000005          SEEK    015642          SETSEC  007705
LINKLN= 000010          NOU     023674          RCVERR= 000400          SEKCNT  016274          SETUP   005134
LNCONT  002670          NOUNIT  003352          RCVRDY= 000001          SEKEXT  016053          SHRTTO= 000000
LNCYL   002662          NUMBB   022634          RDSTAT  001470          SEKINP= 002000          SLOP    023726
LOBYTE= 000377          NUML2S= 000010          READ    011615          SEKOUT  016052          SMASKL  017525
LOCYL   002433          NUMPTR= 000005          RECALB  016635          SEKREQ= 004000          SMASKX  017533
LONG  = 002414          NXTLEV= 010000          RECOVR  004505          SEKTST  016220          SNDAGN  001550
LONGTO= 000001          NXTRCT  015432          REDWRT= 000100          SEND  = 000004          SNDEX   014401
LOSEED  003007          NXTTRK  006543          RETRY = 001000          SEQBE   006060          SNDONE= 002356
LRDTRK  002762          ONEPAT  010465          RETS  = 000001          SEQLP   003377          SORT    017236
LSTMOD  014726          ONEPAX  014501          REVCT   015007          SEQNCR  003375          SORTBB  017403
LSTOVL  003005          ONL     002440          REVEC = 000400          SEQSEK= 000100          SORTBE  017302
LSTTRK  007717          ONLYCL= 000200          REVS  = 000007          SEQTG   007147          SORTTG  017461
L$      005517          OUTO    006364          REVSOK  015135          SERCHN  022420          SPINUP  017204
L2.EOF= 000004          OVE.MN= 001364          RM    = 000004          SERRTY  022417          SS    = 000001
L2.ERS= 000003          OVE.MS= 000000          RNDBE   005650          SERSEC= 023057          ST      002442
L2.OPC= 000000          OVL.MN= 022710          RNDTG   006737          SER0    004630          STACK   001416
L2.RLN= 000002          OVL.MS= 005737          RNDO    005447          SER1    004646          STACYL  002756
L2.SLN= 000001          OVL...= 030647          RONLY = 004000          SER10   005205          START   022654
MASK    005630          OVSTRT= 001362          ROOT    003315          SER11   005220          STATUS= 000007
MAXDBN  022635          OVS.MN= 001040          ROOTLP  003320          SER12   005236          STOP    003370
MAXMSK= 177740          OVS.MS= 046660          RREAL = 013400          SER13   005250          STSERR  016422
MAXMUM  006366          OV... = 031267          RSTOP = 100000          SER14   005263          STSEXT  016633
MAXSEC  003020          PATEXT  005457          RTDS    001417          SER15   005311          STSRES= 000366
```

SYMBOL TABLE

```
ST.C  = 000002          S.LETR= 000005          T2CMD  = 060002          U.COPY= 000057          U.SUBP= 000001
ST.CON= 000002          S.MCNT= 000013          T2DLL  = 060001          U.CSEC= 000024          U.SUBU= 000050
ST.DB = 001000          S.MEGR= 000010          T4BB1  = 060005          U.CTRK= 000020          U.TIMH= 000005
ST.DE = 000200          S.MEGW= 000011          T4BB2  = 060006          U.ECCT= 000032          U.TIML= 000006
ST.DF = 000020          S.PARM= 000000          T4MPRM= 060003          U.ELEV= 000027          U.TSEC= 000023
ST.DR = 000040          S.PAT = 000004          T4MXFR= 060011          U.LCYL= 000067          U.UNUM= 000063
ST.EL = 000010          S.SCHR= 000007          T4SEEK= 060010          U.LGRP= 000071          U.WPRT= 000045
ST.ERR= 000002          S.SDCL= 000002          T4SOFT= 060007          U.MASK= 000025          U.WRIT= 000026
ST.FO = 002000          S.SEEK= 000001          T4UPRM= 060004          U.MBN = 000051          VALBIT= 077177
ST.MOD= 000001          S.TGOF= 000015          UCURSR  022631          U.MLEV= 000031          VALID = 100000
ST.MSK= 000000          S.TGSS= 000017          UDADM4= 001000 G        U.MSEC= 000022          WAITSI= 000012
ST.OA = 000200          S.TRKL= 000006          UDA50  = 000000         U.MSTO= 000011          WBLOCK  010550
ST.PE = 000040          TALK    001670          UDA52  = 000001         U.NEXT= 000000          WBUFLN= 000401
ST.PS = 000002          TGS     022642          ULOP    023601          U.NFUN= 000013          WCHECK= 000010
ST.RE = 000100          TG$     021663          UMASK   022633          U.NSEC= 000021          WCHKAL= 000004
ST.REQ= 000001          THREBE  005705          UNADDR= 140000          U.PARM= 000046          WCONT = 040000
ST.RR = 000100          TLEN.U= 000072          UNITS   023553          U.PAT = 000014          WONLY = 002000
ST.RTY= 000003          TLKHST  001517          UNSSUC= 000175          U.PCTG= 000017          WREAL = 122400
ST.RU = 000001          TMEMPL  010277          UP      006212          U.RBN = 000055          WRITBT  002425
ST.SR = 000020          TO      017167          UPG     007427          U.RCOV= 000047          WRITE   010301
ST.S7 = 000400          TRACK   002767          UPT     007243          U.RRTY= 000010          WSTOP = 140000
ST.UNT= 000000          TRACKS= 000020          UREAD  = 000013         U.RTRY= 000030          XBNCYL= 000021
ST.WE = 000010          TRAV    020027          UTOTST= 060012          U.RVER= 000062          XFERRT= 000000
SUBTRV  020044          TRKGRP= 000003          UWRITE= 000014          U.RWER= 000061          XOPLP0  014455
SUBUNT  022630          TRK$    005507          U.CBN  = 000053         U.RWTO= 000012          XOPLP1  014463
SUNIT   022416          TROOT   023012          U.CCNT= 000015          U.SDIL= 000034          XOPLP2  014502
SUSETL  023715          TRYAGN  005667          U.CCOP= 000060          U.SDIS= 000033          XREAD  = 000002
SWAPBE  017351          TSTNEC  016055          U.CCYL= 000064          U.SDI2= 000035          XWRITE= 000003
S.BADP= 000012          T1MSIZ= 060000          U.CGRP= 000066          U.SRTY= 000007          ZEREDC  001660
S.BESS= 000013

. ABS.  062556      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  37184 WORDS  ( 146 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  71 PAGES
,B:UDAT4.L52/C=[30,30]DMAC52/M,B:UDAT4
```

CROSS REFERENCE TABLE (CREF V04.00)

```
AFTOP    64-2#    73-68    74-145   84-17    85-65
AFTWRT   98-129   100-24#
ALLOCM   95-14    95-17    95-40    96-21    97-2#
ATTN      7-23#   34-18    50-49    52-18    52-25    52-27    54-18    54-77    55-11    119-89   144-108
AVAIL     7-26#   34-18    50-49    52-16    52-23    54-41    119-82   120-56   144-108
BBLOP    144-108  144-108#
BEEXT    74-86    74-138   74-140#
BESDWN   74-53    74-76#
BESUP    74-119   74-139#
BEUSED    9-52#   63-36    63-42    125-42   130-97   134-17   134-43   134-61   137-59   144-108  144-108  144-108
BF.DAT    4-88#
BF.ECC    4-90#
BF.EDC    4-89#   98-114*  105-85   105-112  106-71   106-97   142-179*
BLDBB    144-108  144-108#
BLDUNT   144-108  144-108#
BLKCHK    41-2#   100-89   104-58   105-60   109-93
BREAK      5-3#
BSUSEX   144-108  144-108#
BUFARA   121-87#  144-108
BUFFLG    4-62#    4-71    100-65   103-13   104-56
BUFSIZ    36-15    36-20    37-41#
BUILDP    94-24#  107-58   108-81   116-86   119-75
BULDUM    43-2#    98-60   101-68   142-49
C2DFTL     7-65#   39-96   123-38   124-39   130-151
C2HARD     7-64#  105-123  106-106
CALC      44-2#    80-17    80-29    94-45    114-25   117-16
CALCSC    44-60    45-3#   142-67
CBB2     128-12   134-13   134-28   134-39   134-57   138-14   138-26   138-40   144-108#
CHAINS    45-37#   94-44*   95-7     95-18*   98-79    99-7     100-49   102-6    103-9    104-42   105-41   105-104  106-39   109-40
         109-96   110-37   110-53*  113-6
CHGMOD     7-34#   40-22
CHKBB    137-54   138-2#
CHKBES   130-104  134-2#
CHKCYL   130-101  132-2#
CHKDN     49-121   49-125   57-2#
CHKECC   105-71   106-24#
CHKEDC   104-165  105-24#
CHKTG    137-61   139-2#
CHKUP     49-79    49-118   49-128   58-2#
CHNMAX    45-39#   62-103   62-105   144-108*
CHRRES     7-45#  144-108
CLCLBN    73-27    73-33#
CLCMAX   130-94   136-2#
CMP2      41-15    42-2#    74-52    75-13
CMPDAT   105-102  105-127  106-119  108-60   109-24#
CMPEDC    38-2#    98-113  105-83   106-70   142-178
CMPERR   109-73   109-82   109-90#
CMXEX    109-88   109-95   109-125#
CNTLOP    72-37#   72-41
COMCHR   122-40   123-24#
COMPAR     5-9#
COMPDP   140-10   140-31   140-49   141-2#
COMPLT     7-43#   40-6     40-9     40-10    40-11    40-12    40-13    40-15    144-108
COMPSC   130-93   135-2#
COPBB    144-108  144-108#
COPFIN    98-99    98-107#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
COPLP0  98-92#   98-102
COPLP1  98-96#   98-101
COPLP2  98-104#  98-106
COUNTD  72-38    72-42#
CPYBEX 144-108  144-108#
CR.CLR  40-9#    56-52
CR.DIS  40-13#   47-40
CR.ERR  40-12#  108-66
CR.GCR 123-33   144-108#
CR.GST  40-8#    54-44    56-15   144-108
CR.INR  40-15#  120-39   120-58   142-73
CR.MOD  40-10#   56-62
CR.ONL  40-6#    56-9
CR.RUN 124-32   144-108#
CR.SCR 130-52   144-108#
CR.SEK  40-11#  118-9    142-80
CSCEXT 109-76   109-86#
CURBN   45-5#    80-13*   80-15*   80-57    80-59    80-72*   80-74*   80-76*   80-78*   94-37*   94-38*   94-40*   94-41*   96-42
         96-44    96-73*   96-75*  111-74*  111-78*  112-17   112-17   113-14   113-16*  113-51   113-51   114-22*  114-23*  114-41
        114-41   117-10*  117-11*  117-13*  117-14*  142-65*  142-66*  142-124  142-133  142-187* 142-189*
CVT      5-19#   44-62    142-69
CYL     45-8#    80-19    80-31    94-46*   94-48*   114-27*  114-30*  116-60   116-60   117-17   118-6    118-7    118-24   142-77
        142-78
CYLBN  132-11   132-24   133-2#
CYLSCR  80-18    80-30    80-82#
D$     130-82   134-65   134-66   144-109#
D.LIMT   9-4#    43-15*
D.SCHR   9-5#    43-18*
DATCMP   9-57#   64-14    68-6     68-17    105-98   105-125  106-117  108-58   144-108
DATPRE   6-32#   99-10
DBNCYL   6-41#  136-10   136-18
DCEXT   68-7     68-11    68-16    68-19#
DCLOCK   7-24#   34-33
DCLR    40-25#   56-51*
DCMPAL   9-59#   68-12   144-108
DCMPYS  68-13    68-17#
DCREAD  68-9     68-12#
DCYLS    9-44#   43-8     44-17    96-49    100-122  104-88   130-80   132-35   134-36   138-22   142-205  144-108  144-108
DEBUG    1-56#    2-2      47-3     50-55    51-6     62-35    62-47    98-32   100-33   101-32   104-35   105-35   106-32   108-32
        109-32   111-30   116-36   119-34   120-32   144-108  144-109
DIE      9-10#  123-40   124-41   130-121  130-153  137-68   142-211  144-73   144-108
DINIT    5-12#   59-56
DIREC    9-17#   49-14    49-116   57-23    58-35    74-33    80-54    80-70    85-35
DIS     40-13    40-20#
DISCON   7-32#   40-20
DONE     7-18#   47-52
DOWNG   85-41    89-2#
DOWNT   85-39    87-2#    89-23
DRC     40-9     40-24#
DRINIT   9-32#   54-28    54-30    59-58
DROP     9-15#   47-30    47-39    49-28    49-84    49-109   49-132   54-6     55-7     55-35    55-43    57-17    58-18    [illegible]
        130-126  130-142  137-73   137-89
DRPALL  55-37   121-24#
DRTYPE   6-15#
DRVCLR   7-37#   40-24
DRVEXT 144-74   144-89#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
FNDLOP  75-11#   75-15
FNDRER 102-25   103-2#
FORMAT   5-4#
FSTOP    4-65#
FT.BUF   4-27#
FT.HI    4-28#
FT.LOW   4-29#
FTIME    9-20#   49-11    49-14    55-47    58-33   144-108  144-108
FTLDEV   7-61#    7-65     7-65     34-9     52-20    54-3     54-5     56-25    99-15    99-44   102-11   116-59   133-22   134-21
       134-23   134-30   134-53   134-67   134-69   137-40   138-18   138-36   138-42   139-45   140-29   142-207  144-71   144-108
       144-108
FTLSYS   7-60#
FWRD    109-60   109-89#
GCR     144-108  144-108#
GETCHR    7-38#  144-108
GETMEM  130-71   144-108  144-108  144-108#
GETSER  130-148  137-95   142-24#
GETSTA    7-40#   40-26
GETSUB    7-39#  144-108
GETU    144-108  144-108#
GMPARM  144-108  144-108#
GO4IT    50-43#
GOINIT   39-23    39-55    59-48#  100-115  100-155  100-160  104-81   104-133  104-138
GORCLB   56-40    59-39#  100-146  104-124  116-65
GORTRY   39-35    39-88    50-51    59-21#  120-54
GOSEEK   49-13    56-50    59-30#   59-57    62-118   99-17    99-46   100-176  102-13   104-154  119-111
GOTOBE   72-47    72-50    72-53#
GOTOFS   72-43    72-56#
GROUP    45-9#    94-50*  116-60   118-8    142-79
GRP$    139-33   144-109#
GRPCYL    6-24#  135-15   139-32   140-42
GRPOFF    6-37#
GST      40-8     40-26#
HBHINB    6-49#   42-11    44-38    44-50    74-68    74-96    77-26    78-27    79-11   109-55   111-65   113-13   126-16   133-20
       139-22   144-108  144-108  144-108  144-108
HBLONB    6-50#   44-29    44-33    44-58    98-77   142-131
HD.BAD    4-81#
HD.DBN    4-84#   96-51
HD.LBN    4-78#   96-55
HD.PRV    4-82#
HD.RBN    4-79#   96-53
HD.REV    4-80#
HD.XBN    4-83#  142-132
HDRPRE    6-33#
HIBN     44-34*   44-59*   45-2#    54-32    59-54*   96-54
HIBYTE    6-47#   43-12    44-11    44-13    44-20    44-44    54-81    56-46    83-12    85-59    92-8     99-11   123-53   123-57
       127-18   128-27   129-14   130-84   130-90   135-9    135-16   136-12   136-19   139-37   139-41   140-9    140-25   140-46
       140-47   142-54   142-56
HICYL     6-23#   44-37    44-49   114-27
HIDBN     6-29#   44-53
HILBN     6-25#   44-32
HIMEM     4-18#    8-79    45-26
HIRBN     6-28#   44-28
HISEED   45-24#   70-9     70-19    70-25*   73-23    81-40
HIXBN     6-26#  142-126
HOSTRQ   35-7     36-2#    39-106   47-53    49-77    53-6     55-4     55-13    99-51   100-144  102-30   104-122  105-92   106-86
```

	106-89	116-68	119-63	119-70	119-91	121-57	130-120	137-47	137-67	142-102	142-210	142-221	144-108	144-108
	144-108	144-108	144-108	144-108	144-108	144-108	144-108							
HRDREV	6-13#													
HRQ.01	34-10*	35-7*	37-6#	39-24*	39-40*	39-56*	39-60*	39-64*	39-68*	39-72*	39-75*	39-82*	39-106*	49-77*
	49-77*	52-20*	52-28*	53-3*	53-4*	53-7*	55-5*	55-13*	56-25*	98-50*	99-15*	99-44*	99-51*	100-114*
	100-129*	100-144*	100-145*	100-150*	100-154*	100-159*	100-164*	100-166*	101-53*	102-11*	102-30*	104-80*	104-96*	104-114*
	104-118*	104-122*	104-123*	104-128*	104-132*	104-137*	104-142*	104-144*	105-73*	105-92*	105-107*	106-53*	106-59*	106-84*
	106-89*	106-89*	106-92*	108-48*	109-97*	112-10*	113-44*	114-34*	116-59*	116-68*	119-60*	119-61*	119-70*	119-71*
	119-91*	119-92*	119-108*	121-49*	133-22*	134-21*	134-23*	134-30*	134-53*	134-67*	134-69*	137-40*	137-47*	137-47*
	138-18*	138-36*	138-42*	139-45*	140-29*	142-100*	142-207*	142-212*	144-71*	144-108	144-108	144-108	144-108	144-108
	144-108*	144-108*	144-108*	144-108*	144-108*	144-108*	144-108*	144-108*	144-108*					
HRQ.02	34-9*	35-7*	37-7#	39-24*	39-40*	39-56*	39-60*	39-64*	39-68*	39-72*	39-75*	39-82*	39-96*	39-106*
	49-77*	52-20*	52-28*	54-1	55-13*	56-25*	98-50*	98-123*	99-15*	99-44*	99-51*	100-114*	100-129*	100-144*
	100-145*	100-150*	100-154*	100-159*	100-164*	100-166*	101-53*	102-11*	102-30*	104-80*	104-96*	104-114*	104-118*	104-122*
	104-123*	104-128*	104-132*	104-137*	104-142*	104-144*	105-73*	105-92*	105-107*	105-123*	106-53*	106-59*	106-77*	106-80*
	106-89*	106-92*	106-106*	108-48*	109-97*	110-64*	112-10*	113-44*	114-34*	116-59*	116-68*	119-70*	119-71*	119-91*
	119-92*	119-108*	121-37*	123-38*	124-39*	130-151*	133-22*	134-21*	134-23*	134-30*	134-53*	134-67*	134-69*	137-40*
	137-47*	138-18*	138-36*	138-42*	139-45*	140-29*	142-97*	142-207*	142-213*	144-71*	144-108	144-108	144-108*	144-108*
HRQ.03	35-7*	37-8#	39-106*	55-1*	55-2*	55-13*	98-124*	99-51*	100-144*	102-30*	104-122*	105-92*	106-73*	106-89*
	110-65*	116-68*	119-70*	119-91*	121-55*	130-117*	130-118*	137-64*	137-65*	142-98*	142-208*	142-214*	144-108	144-108
	144-108	144-108*	144-108*											
HRQ.04	34-6*	34-8*	35-7*	37-9#	39-24*	39-40*	39-56*	39-60*	39-64*	39-68*	39-72*	39-75*	39-82*	39-106*
	52-20*	52-28*	55-13*	56-25*	98-50*	99-15*	99-44*	99-51*	100-114*	100-129*	100-144*	100-145*	100-150*	100-154*
	100-159*	100-164*	100-166*	101-53*	102-11*	102-30*	104-80*	104-96*	104-114*	104-118*	104-122*	104-123*	104-128*	104-132*
	104-137*	104-142*	104-144*	105-73*	105-92*	105-107*	106-53*	106-59*	106-89*	106-92*	108-48*	109-97*	112-10*	113-44*
	114-34*	116-59*	116-68*	119-70*	119-71*	119-91*	119-92*	119-108*	121-38	133-22*	134-21*	134-23*	134-30*	134-53*
	134-67*	134-69*	137-40*	138-18*	138-36*	138-42*	139-45*	140-29*	142-99*	142-207*	142-215*	144-71*	144-108*	144-108*
HRQ.05	37-10#	52-29*	56-13*	56-19*	56-28*	56-32*	56-34*	56-56*	56-64*	98-50*	99-15*	99-44*	100-130*	100-166*
	100-169*	101-53*	102-11*	104-97*	104-144*	104-147*	105-73*	105-107*	106-60*	106-89*	106-92*	108-48*	108-70*	109-98*
	109-101*	112-14*	112-16*	113-48*	113-50*	114-38*	114-40*	116-59*	118-33*	119-71*	119-92*	119-108*	120-65*	123-37*
	124-38*	130-150*	133-22*	134-21*	134-23*	134-30*	134-53*	134-67*	134-69*	138-18*	138-36*	138-42*	139-45*	140-29*
	142-207*	142-216*	144-71*	144-108	144-108*									
HRQ.06	37-11#	39-73*	39-76*	39-83*	56-35*	98-50*	99-15*	99-45*	100-130*	100-166*	100-170*	101-53*	102-11*	104-97*
	104-144*	104-148*	105-73*	105-107*	106-60*	106-89*	106-92*	108-48*	109-102*	112-17*	113-51*	114-41*	116-59*	119-71*
	119-92*	119-108*	134-47*	134-53*	134-65*	134-69*	138-36*	138-42*	139-45*	140-29*	142-217*			
HRQ.07	37-12#	39-77*	39-83*	98-50*	99-15*	100-130*	100-167*	100-170*	101-53*	102-11*	104-97*	104-145*	104-148*	105-73*
	105-107*	106-60*	106-89*	106-92*	108-48*	109-102*	112-17*	113-51*	114-41*	116-59*	119-71*	119-92*	119-108*	134-48*
	134-53*	134-66*	134-69*	138-36*	138-42*	139-45*	142-218*	144-108	144-108					
HRQ.08	37-13#	39-83*	98-51*	99-16*	100-131*	100-170*	101-54*	102-12*	104-98*	104-148*	105-74*	105-108*	106-61*	106-89*
	106-93*	108-49*	109-102*	112-17*	114-41*	116-60*	119-71*	119-92*	119-109*	142-219*				
HRQ.09	37-14#	98-51*	100-131*	100-170*	101-54*	104-98*	104-148*	105-74*	105-108*	106-61*	106-89*	106-93*	108-49*	109-103*
	112-17*	114-41*	116-60*	119-72*	119-92*	119-109*								
HRQ.10	37-15#	98-51*	100-131*	100-171*	101-54*	104-98*	104-149*	105-74*	105-108*	106-61*	106-89*	106-93*	108-49*	109-103*
	116-60*	119-72*	119-93*	119-109*										
HRQ.11	37-16#	98-51*	100-132*	100-171*	101-54*	104-99*	104-149*	105-75*	105-109*	106-62*	106-94*	108-49*	109-104*	116-60*
	119-72*	119-94*	119-110*											
HRQ.12	37-17#	100-132*	100-171*	104-99*	104-149*	105-75*	105-109*	106-62*	106-94*	108-49*	109-104*	116-60*	119-72*	
HRQ.13	37-18#	100-133*	100-172*	104-100*	104-150*	105-76*	105-110*	106-63*	106-95*	108-50*	109-104*	116-60*	119-72*	
HRQ.14	37-19#	100-133*	100-172*	104-100*	104-150*	105-76*	105-110*	106-63*	106-95*	108-50*	109-105*	116-60*	119-72*	
HRQ.15	37-20#	100-133*	100-173*	104-100*	104-151*	105-76*	105-110*	106-63*	106-95*	109-105*	119-73*			
HRQ.16	37-21#	100-133*	100-173*	104-100*	104-151*	105-77*	105-111*	106-64*	106-96*	109-105*				
HRQ.17	37-22#	100-134*	100-173*	104-101*	104-151*	105-77*	105-111*	106-64*	106-96*	109-105*				
HRQ.18	37-23#	100-134*	100-173*	104-101*	104-151*	105-77*	105-111*	106-64*	106-96*	109-122*				
HRQ.19	37-24#	100-134*	100-174*	104-101*	104-152*	105-78*	105-112*	106-65*	106-97*	109-122*				
HRQ.20	37-25#	100-174*	104-152*	105-112*	106-97*	109-122*								
HRQ.21	37-26#	100-174*	104-152*	105-112*	106-97*	109-122*								

```
HRQ.22  37-27#  100-175* 104-153* 109-122*
HRQ.23  37-28#  109-122*
HRQ.24  37-29#  109-122*
HRQ.25  37-30#  109-122*
HRQ.26  37-31#  109-122*
HRQ.27  37-32#  109-122*
HRQ.28  37-33#  109-122*
HRQ.29  37-34#  109-123*
HRQ.30  37-35#  109-123*
HRQ.31  37-36#  109-123*
HRQ.32  37-37#  109-123*
HRQ.33  37-38#
HRQ.34  37-39#
HRQ.RQ  34-11*   36-13*   36-14    36-19    36-22*   37-5#    37-41    39-24*   39-40*   39-56*   39-60*   39-64*   39-68*   39-72*
        39-75*   39-82*   52-20*   52-28*   53-1*    53-5     54-10    55-3     56-25*   98-50*   98-126*  99-15*   99-44*   100-114*
       100-129* 100-145* 100-150* 100-154* 100-159* 100-164* 100-166* 101-53*  102-11*  104-80*  104-96*  104-114* 104-118* 104-123*
       104-128* 104-132* 104-137* 104-142* 104-144* 105-73*  105-107* 105-124* 106-53*  106-59*  106-92*  106-107* 108-48*  109-97*
       110-67*  112-10*  113-44*  114-34*  116-59*  119-71*  119-92*  119-108* 123-39*  124-40*  130-119  130-152* 133-22*  134-21*
       134-23*  134-30*  134-53*  134-67*  134-69*  137-40*  137-66   138-18*  138-36*  138-42*  139-45*  140-29*  142-207* 142-209
       144-71*  144-108  144-108  144-108* 144-108*
ID      80-27    80-47    80-69#
INCODT   1-57#    1-68    33-21    59-66
INDEX   45-12#   96-57
INITD  142-229  144-24#
INITW    9-16#   47-34    49-32    49-74    49-85    58-22    58-32    62-72    63-34    65-8     67-10    88-11    137-42   137-44
       144-45   144-51   144-108  144-108
INR     40-15    40-27#
INS     40-11    40-28#   118-6*   118-7*   118-8*
INSEEK   7-42#   40-28
INSET  122-24#  144-108
INTEDC   8-77#   38-9
INTINP   9-11#   55-38    144-108  144-108
IRECLB   7-41#   40-27
ISUEXT 118-32   118-34#
ISUSEK 116-72   118-2#
IWIPRG   9-9#    47-33    47-49    47-50    49-31    49-73    58-21    58-31    76-9     80-11    85-53    144-53
JMPRET   50-52   51-4#    62-122   63-49    69-10    73-71    74-148   84-20    85-68    96-87    98-131   100-181  101-72   104-168
       105-133  106-123  107-67   108-85   109-127  110-71   111-89   116-89   119-112  120-77   122-41   123-84   124-43   125-55
       130-157  137-100  142-231  144-89
L$      115-18   130-86   134-47   134-48   144-109#
L2.EOF   7-56#   39-90    39-101   120-58
L2.ERS   7-55#    7-56    39-80    39-83
L2.OPC   7-52#    7-53    39-14    54-47    144-108
L2.RLN   7-54#    7-55    39-43    54-57    144-108
L2.SLN   7-53#    7-54    39-15    54-48    144-108
LASTU  144-108# 144-108* 144-108* 144-108*
LBHINB   6-51#   56-43    144-84
LBLONB   6-52#   66-10    66-13    80-49    98-77    109-43   109-62   123-50   123-60   123-64   123-78   144-108
LBNCYL   6-22#  114-30   130-107  130-110  142-51   142-52
LBNHST   6-38#  111-60   111-62   132-37   132-40   134-38   134-50*  134-52*  134-53   134-53   138-25
LBNTRK   6-36#   44-12    44-19    44-43   130-89   142-55
LEVNZR 108-44   108-46   108-65#
LEVUSD   9-33#   62-116   62-119   108-76   119-77
LINKLN   4-104#  95-16    95-39    144-108  144-108  144-108
LNCONT  44-30    44-34#
LNCYL   44-23    44-31#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

56-19#	56-25#	56-28	56-28	56-28#	56-28#	56-32	56-32	56-32#	56-32#	56-34	56-34	56-34#	56-34#
56-35	56-56	56-56	56-56#	56-56#	56-64	56-64	56-64#	56-64#	98-50	98-50	98-50	98-50	98-50
98-50	98-50#	98-50#	98-50#	98-50#	98-51	98-51	98-51	98-51	98-51	98-51	98-51	98-51	98-51#
98-51#	98-51#	98-51#	99-15	99-15	99-15	99-15	99-15	99-15	99-15#	99-15#	99-15#	99-15#	99-16
99-44	99-44	99-44#	99-44#	99-45	100-114#	100-129#	100-130	100-130	100-130	100-130	100-130	100-130	100-130#
100-130#	100-130#	100-131	100-131	100-131	100-131	100-131	100-131	100-131#	100-131#	100-131#	100-132	100-132	100-132
100-132	100-132#	100-132#	100-133	100-133	100-133	100-133	100-133	100-133	100-133	100-133	100-133#	100-133#	100-133#
100-133#	100-134	100-134	100-134	100-134	100-134	100-134	100-134#	100-134#	100-134#	100-145#	100-150#	100-154#	100-159#
100-164#	100-166	100-166	100-166	100-166	100-166#	100-166#	100-166#	100-167	100-169	100-169	100-169#	100-169#	100-170
100-170	100-170	100-170	100-170	100-170	100-170	100-170	100-170#	100-170#	100-170#	100-170#	100-171	100-171	100-171
100-171	100-171	100-171	100-171#	100-171#	100-171#	100-172	100-172	100-172	100-172	100-172#	100-172#	100-173	100-173
100-173	100-173	100-173	100-173	100-173	100-173	100-173#	100-173#	100-173#	100-173#	100-174	100-174	100-174	100-174
100-174	100-174	100-174#	100-174#	100-174#	100-175	101-53	101-53	101-53	101-53	101-53	101-53	101-53#	101-53#
101-53#	101-53#	101-54	101-54	101-54	101-54	101-54	101-54	101-54	101-54	101-54#	101-54#	101-54#	101-54#
102-11	102-11	102-11	102-11	102-11	102-11	102-11#	102-11#	102-11#	102-11#	102-12	104-80#	104-96#	104-97
104-97	104-97	104-97	104-97	104-97	104-97#	104-97#	104-97#	104-98	104-98	104-98	104-98	104-98	104-98
104-98#	104-98#	104-98#	104-99	104-99	104-99	104-99	104-99#	104-99#	104-100	104-100	104-100	104-100	104-100
104-100	104-100	104-100	104-100#	104-100#	104-100#	104-100#	104-101	104-101	104-101	104-101	104-101	104-101	104-101#
104-101#	104-101#	104-114#	104-118#	104-123#	104-128#	104-132#	104-137#	104-142#	104-144	104-144	104-144	104-144	104-144#
104-144#	104-144#	104-145	104-147	104-147	104-147#	104-147#	104-148	104-148	104-148	104-148	104-148	104-148	104-148
104-148	104-148#	104-148#	104-148#	104-148#	104-149	104-149	104-149	104-149	104-149	104-149	104-149#	104-149#	104-149#
104-150	104-150	104-150	104-150	104-150#	104-150#	104-151	104-151	104-151	104-151	104-151	104-151	104-151	104-151
104-151#	104-151#	104-151#	104-151#	104-152	104-152	104-152	104-152	104-152	104-152	104-152#	104-152#	104-152#	104-153
105-73	105-73	105-73	105-73	105-73	105-73	105-73#	105-73#	105-73#	105-73#	105-74	105-74	105-74	105-74
105-74	105-74	105-74#	105-74#	105-74#	105-75	105-75	105-75	105-75	105-75#	105-75#	105-76	105-76	105-76
105-76	105-76	105-76	105-76#	105-76#	105-76#	105-77	105-77	105-77	105-77	105-77	105-77	105-77#	105-77#
105-77#	105-78	105-78	105-78#	105-107	105-107	105-107	105-107	105-107	105-107	105-107#	105-107#	105-107#	105-107#
105-108	105-108	105-108	105-108	105-108	105-108	105-108#	105-108#	105-108#	105-109	105-109	105-109	105-109	105-109#
105-109#	105-110	105-110	105-110	105-110	105-110	105-110	105-110#	105-110#	105-110#	105-111	105-111	105-111	105-111
105-111	105-111	105-111#	105-111#	105-111#	105-112	105-112	105-112	105-112	105-112	105-112	105-112#	105-112#	105-112#
106-53#	106-59#	106-60	106-60	106-60	106-60	106-60	106-60	106-60#	106-60#	106-60#	106-61	106-61	106-61
106-61	106-61	106-61	106-61#	106-61#	106-61#	106-62	106-62	106-62	106-62	106-62#	106-62#	106-63	106-63
106-63	106-63	106-63	106-63	106-63#	106-63#	106-63#	106-64	106-64	106-64	106-64	106-64	106-64	106-64#
106-64#	106-64#	106-65	106-65	106-65#	106-89	106-89	106-89	106-89	106-89	106-89	106-89	106-89	106-89
106-89	106-89	106-89	106-89	106-89	106-89	106-89	106-89#	106-89#	106-89#	106-89#	106-89#	106-89#	106-89#
106-89#	106-89#	106-92	106-92	106-92	106-92	106-92	106-92	106-92#	106-92#	106-92#	106-92#	106-93	106-93
106-93	106-93	106-93	106-93	106-93#	106-93#	106-93#	106-94	106-94	106-94	106-94	106-94#	106-94#	106-95
106-95	106-95	106-95	106-95	106-95	106-95#	106-95#	106-95#	106-96	106-96	106-96	106-96	106-96	106-96
106-96#	106-96#	106-96#	106-97	106-97	106-97	106-97	106-97	106-97	106-9[illegible]#	106-97#	106-97#	108-48	108-48
108-48	108-48	108-48	108-48	108-48#	108-48#	108-48#	108-48#	108-49	108-49	108-49	108-49	108-49	108-49
108-49	108-49	108-49	108-49	108-49#	108-49#	108-49#	108-49#	108-49#	108-50	108-50	108-50	108-50	108-50#
108-50#	108-70	108-70	108-70#	108-70#	109-97#	109-98	109-98	109-98#	109-98#	109-101	109-101	109-101#	109-101#
109-102	109-102	109-102	109-102	109-102	109-102	109-102#	109-102#	109-102#	109-103	109-103	109-103	109-103	109-103#
109-103#	109-104	109-104	109-104	109-104	109-104	109-104	109-104#	109-104#	109-104#	109-105	109-105	109-105	109-105
109-105	109-105	109-105	109-105	109-105#	109-105#	109-105#	109-105#	109-122	109-122	109-122	109-122	109-122	109-122
109-122	109-122	109-122	109-122	109-122	109-122	109-122	109-122	109-122	109-122	109-122	109-122	109-122	109-122
109-122	109-122	109-122#	109-122#	109-122#	109-122#	109-122#	109-122#	109-122#	109-122#	109-122#	109-122#	109-122#	109-123
109-123	109-123	109-123	109-123	109-123	109-123	109-123	109-123#	109-123#	109-123#	109-123#	112-10#	112-14	112-14
112-14#	112-14#	112-16	112-16	112-16#	112-16#	112-17	112-17	112-17	112-17	112 17	112-17	112-17	112-17
112-17#	112-17#	112-17#	112-17#	113-44#	113-48	113-48	113-48#	113-48#	113-50	113-50	113-50#	113-50#	113-51
113-51	113-51	113-51	113-51#	113-51#	114-34#	114-38	114-38	114-38#	114-38#	114-40	114-40	114-40#	114-40#
114-41	114-41	114-41	114-41	114-41	114-41	114-41	114-41	114-41#	114-41#	114-41#	114-41#	116-59	116-59
116-59	116-59	116-59	116-59	116-59#	116-59#	116-59#	116-59#	116-60	116-60	116-60	116-60	116-60	116-60
116-60	116-60	116-60	116-60	116-60	116-60	116-60	116-60	116-60#	116-60#	116-60#	116-60#	116-60#	116-60#
116-60#	118-33	118-33	118-33#	118-33#	119-71	119-71	119-71	119-71	119-71	119-71	119-71	119-71	119-71#
119-71#	119-71#	119-71#	119-71#	119-72	119-72	119-72	119-72	119-72	119-72	119-72	119-72	119-72	119-72

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
        109-102  111-49*  115-19*  116-59   119-71   130-88*
S.MCNT    8-70#    8-71    81-37    81-42    81-45    81-48    90-7     90-9     91-10    91-12
S.MEGR    8-62#    8-63   104-64   104-65*  104-158  104-160* 105-62   105-64*  107-39   107-40*  110-60   110-68*  130-58*
S.MEGW    8-63#    8-64    98-118   98-122*  100-79   100-80*  130-59*
S.PARM    8-55#    8-56    43-7     44-16    49-26    49-62    49-76    49-104   55-42    55-45    57-15    58-16    62-69    63-33
          65-7     83-7    85-34    85-50    96-48    98-47   100-121  101-47   104-87   105-68   105-118  106-100  125-41   130-79
         130-96   130-128  130-140  132-34   134-16   134-35   134-42   134-60   137-35   137-46   137-58   137-75   137-87   138-21
         139-28   140-38   141-9    142-204  144-39   144-63   144-108  144-108
S.PAT     8-58#    8-59    66-8     144-108*
S.SCHR    8-61#    8-62    43-10    43-16    44-9     44-61    83-10    85-57    92-6     99-9     111-52   112-22   114-11   114-26
         130-72*  131-18   131-39*  138-24   139-29   140-39   142-50   142-64   142-68   142-125  142-186  144-108*
S.SDCL    8-57#    8-58    44-47    44-51   130-109* 130-113*
S.SEEK    8-56#    8-57   119-57   119-66*  142-41*
S.TGOF    8-71#    8-72    82-5     90-11    90-13    91-8     91-9    139-21   139-23*  140-6    140-59   140-60
S.TGSS    8-72#   82-15    86-27    87-22    90-15    91-14    91-21   129-6    139-13   140-64   140-70   144-108  144-108
S.TRKL    8-60#    8-61    43-13    76-14    76-16    83-5     85-48    88-13    89-16    93-10   130-92*
SBCRES    7-46#  144-108
SCHAR0  125-54   130-24#  130-154  137-97
SCHAR1  130-115  137-24#
SCHRCP  131-20   131-30#
SCR     144-108  144-108#
SCR1      44-14*   45-16#  49-136*  62-77*   62-80*   74-35*   74-87*   76-10*   86-8*    87-6*    88-6*    89-6*    96-59   144-42*
         144-44*  144-66*  144-69*
SCR2      45-17#  101-66*  104-110* 105-93*  105-105* 105-112  105-114* 106-41*  106-88*  108-53*  109-99*  111-79
SCTWRD    8-76#   98-91   109-75   109-84
SDI$      40-5#    40-6     40-6     40-6#    40-8     40-8     40-8#    40-9     40-9     40-9#    40-10    40-10    40-10#   40-11
          40-11    40-11#   40-12    40-12    40-12#   40-13    40-13    40-13#   40-15    40-15    40-15#   40-16   144-108  144-108
         144-108  144-108  144-108  144-108  144-108  144-108# 144-108# 144-108# 144-108#
.DIVER    6-4#
SEARCH  111-83   113-2#
SEC512    4-72#   99-24   102-19   106-45   142-146
SEC576    4-73#
SECCHK  101-70   104-24#  110-54
SECCYL  133-17   135-20*  136-13   144-108#
SECGRP  135-13*  135-17   136-20   141-13   144-108#
SECMAX    45-38#  62-99    62-101   144-108*
SECPTR    95-15*  96-14    97-11#
SECTOR    45-11#  96-40
SECTRK  130-91*  135-10   141-15   144-108#
SEEK      55-24   62-114  111-70   112-33   114-47   116-24#  119-87   120-76
SEKCNT  119-59   119-66#
SEKEXT  116-67   116-74   116-85   116-89#
SEKINP    9-19#  116-44   116-63   118-30   119-52   119-85
SEKOUT  116-53   116-88#
SEKREQ    9-35#   49-44    55-22    59-35    59-44   117-19   119-55   120-69
SEKTST  116-83   119-24#  119-44   119-99
SEND      5-7#    39-16    54-49   144-108
SEQBE     63-44   74-24#
SEQLP     48-11#  55-53
SEQNCR    47-36   48-2#   144-108
SEQSEK    9-51#   49-85    62-70    63-34   144-40   144-64   144-108
SEQTG     63-46   85-24#
SER0      56-19  144-109#
SER1      56-56  144-109#
SER10   144-108  144-109#
SER11   144-108  144-109#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
SER12   144-108  144-109#
SER13   144-108  144-109#
SER14   144-108  144-109#
SER15   144-108  144-109#
SER16   144-108  144-109#
SER17   144-108  144-109#
SER18   144-108  144-109#
SER18A  121-64   144-108  144-109#
SER18B  121-65   144-108  144-109#
SER18C  121-66   144-108  144-109#
SER18D  121-67   144-108  144-109#
SER18E  144-108  144-108#
SER18F  121-54   121-64#
SER19   100-169  104-147  144-109#
SER2     56-13   144-109#
SER20   133-22   134-21   134-23   134-30   134-53   134-67   134-69   138-18   138-36   138-42   139-45   140-29   144-71   144-109#
SER21   105-73   105-107  106-60   106-92   144-109#
SER22    39-73    39-77    39-83    52-29    56-35    99-16    99-45   100-167  100-175  102-12   104-145  104-153  119-94   119-110
        144-109#
SER24   109-98   144-109#
SER25   109-101  144-109#
SER26   112-14   113-48   114-38   144-109#
SER3     56-64   144-109#
SER32   112-16   113-50   114-40   144-109#
SER36   100-166  104-144  144-109#
SER39   144-108  144-109#
SER4    108-70   144-109#
SER40   144-108  144-109#
SER41   144-108  144-109#
SER42    56-28   144-109#
SER43    56-32   144-109#
SER44    56-34   144-109#
SER45   144-108  144-109#
SER5    130-150  144-109#
SER6    124-38   144-109#
SER7    120-65   144-109#
SER8    123-37   144-109#
SER9    118-33   144-109#
SERCHN  142-120* 142-124* 142-134* 142-135* 142-136* 142-137* 142-139  142-163* 142-168  143-4#
SERRTY  142-61*  142-191* 143-3#
SERSEC  142-177  142-179* 142-196* 142-197* 142-198* 142-199* 142-201* 142-207  142-216  142-217  142-218  142-219  143-14   144-108#
SETSEC   88-9     88-25    92-2#    93-6
SETUP    55-49    62-24#   69-9     98-58   100-86   100-94   100-140  101-61   104-107  110-69   115-20   144-86
SHRTTO    6-3#   123-63
SLOP    144-108  144-108#
SMASKL  130-46#  130-49
SMASKX  130-46   130-50#
SNDAGN   36-14#   36-18
SNDEX   108-71   108-85#
SNDONE   39-10    40-7#
SORT    124-36   125-24#
SORTBB  125-47   128-2#
SORTBE  125-46   126-2#
SORTTG  125-44   129-2#
SPINUP  123-81   124-24#
SS        6-9#
```

```
ST       39-42    40-38#   54-56    54-71    54-80    54-84*   56-22    56-41    56-45    123-42   123-43   123-44   123-45   123-52
        123-55   123-59   123-63   123-76   130-60   130-83   130-89   130-107  130-108  130-110  131-31   132-37   132-40   134-38
        134-50*  134-52*  134-53   134-53   135-8    135-15   136-10   136-18   144-108  144-108
ST.C     5-37#
ST.CON   5-36#
ST.DB    5-57#
ST.DE    5-51#    56-48
ST.DF    5-54#
ST.DR    5-46#
ST.EL    5-48#    54-84
ST.ERR   5-35#    54-80    56-45
ST.FO    5-56#
ST.MOD   5-34#    56-41
ST.MSK   5-32#
ST.OA    5-44#
ST.PE    5-53#
ST.PS    5-49#    56-23
ST.RE    5-52#
ST.REQ   5-33#    54-84*   56-22
ST.RR    5-45#    56-38
ST.RTY   5-38#
ST.RU    5-50#    56-23    56-26
ST.S7    5-58#
ST.SR    5-47#    56-23    56-30
ST.UNT   5-31#
ST.WE    5-55#
STACK   33-19     33-32#
STACYL  44-36*    44-39*   44-48*   44-52*   45-4#    142-51*  142-52*
START   33-27    144-108#
STATUS   5-10#    34-32
STOP    47-48     47-52#   144-108  144-108
STSERR 119-49    119-82#
STSEXT 119-46    119-81   119-88   119-95   119-103  119-107  119-112#
STSRES   7-47#    40-8
SUBTRV 131-3     131-13#
SUBUNT 130-51*   144-108#
SUNIT  142-33*   142-100  142-208  142-212  142-223* 142-224* 143-2#
SUSETL 144-108   144-108#
SWAPBE 126-23    127-2#
T1MSIZ   7-4#
T2CMD    7-6#
T2DLL    7-5#
T4BB1    7-9#    144-108  144-108
T4BB2    7-10#   144-108
T4MPRM   7-7#    144-108
T4MXFR   7-13#   53-1     98-125   110-66
T4SEEK   7-12#   119-62
T4SOFT   7-11#   35-7     39-106   55-13    99-51    100-144  102-30   104-122  105-92   106-85   116-68   119-70   119-91   142-101
T4UPRM   7-8#    142-220  144-108  144-108
TALK    39-2#    47-41    56-10    56-16    56-53    56-63    108-67   118-10   120-40   123-34   124-33   130-54   142-74   142-81
TG$     139-33*  139-36*  139-45   140-29   140-91#
TGS     139-12   139-38   140-5    140-45   144-108#
THREBE   72-48    72-51#
TLEN.U   8-78#    8-79     144-108
TLKHST   35-2#    39-50    94-32
TMEMPL   94-42*   97-6*    97-7     97-10#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
U.RCOV    8-38#   8-39    49-45*  50-43   52-10   54-27   54-31*  55-8    55-18   55-21*  59-7    59-9*   59-16   59-18*
         59-27*  59-36*  59-45*  59-59*  62-120* 107-62  107-64* 108-39  108-72  108-77* 116-48  117-18  119-54  119-56*
        119-76  120-67  120-70* 144-108*
U.RRTY    8-15#   8-16    62-113* 105-73  105-87  105-107 106-60  106-74  106-92  107-33  107-55* 108-83*
U.RTRY    8-30#   8-31    107-52  123-51*
U.RVER    8-46#   8-47    111-51* 112-11  113-45  114-35
U.RWER    8-45#   8-46    99-30*  99-44   100-81  100-109 102-24* 104-75  111-50
U.RWTO    8-17#   8-18    62-110* 98-40   98-42*  99-49   100-170 101-40  101-42* 102-28  104-148 107-57* 111-69* 112-32*
        114-46*
U.SDI2    8-35#   8-36    40-6    40-8    40-9    40-10   40-11   40-12   40-13   40-15   49-48   144-108 144-108 144-108
U.SDIL    8-34#   8-35    39-30   123-62*
U.SDIS    8-33#   8-34    39-27   54-55   123-66* 144-108*
U.SRTY    8-14#   8-15    116-47* 116-54  116-75  116-77* 119-68
U.SUBP    8-11#   8-12    49-18   49-41   49-57   49-92   57-6    58-6    121-41  125-35  130-122 130-135 131-15  137-69
        137-82  142-36  144-34  144-55  144-108 144-108
U.SUBU    8-39#   8-40    35-7    39-106  49-38*  49-58   49-77   49-78   49-94   49-112* 49-115  53-4    55-2    55-13
         57-18*  58-24*  99-37   99-51   100-144 102-30  104-122 105-92  106-83  106-89  116-68  119-61  119-70  119-91
        130-35  130-118 130-124 130-130 130-132* 130-145* 137-47  137-65  137-71  137-77  137-79* 137-92*
U.TIMH    8-12#   8-13    116-79* 119-104 119-106* 142-85* 142-107 142-109*
U.TIML    8-13#   8-14    116-81* 119-100 119-102* 142-87* 142-103 142-105*
U.TSEC    8-25#   8-26    49-22*  49-70*  62-45   62-78   62-90   73-62*  74-77   74-109  76-17*  80-65*  84-16*  85-64*
U.UNUM    8-47#   8-48    35-7    39-106  49-77   53-3    55-1    55-13   99-51   100-144 102-30  104-122 105-92  106-82
        106-89  116-68  119-60  119-70  119-91  121-46  130-117 137-47  137-64  142-33  142-34  144-108 144-108*
U.WPRT    8-36#   8-37    56-59   99-36   144-85* 144-108*
U.WRIT    8-28#   8-29    56-44*  56-58   99-32
UCURSR  144-108 144-108 144-108 144-108 144-108 144-108# 144-108* 144-108* 144-108*
UDA50     1-40#   1-48    1-63    1-75    4-10    4-41    4-97    6-54    8-81    33-5    41-17   49-142  60-6    62-3
         62-17   62-125  63-3    63-17   71-12   71-18   72-3    72-17   73-26   73-32   73-74   74-3    74-17   80-87
         81-3    81-17   81-43   81-49   82-12   83-25   84-15   84-23   85-3    85-17   91-18   93-20   94-3    94-17
         95-22   95-45   96-26   97-13   98-3    98-17   98-45   99-55   100-3   100-17  100-54  100-90  100-184 101-3
        101-17  101-45  103-27  104-3   104-17  104-45  104-171 105-3   105-17  105-43  105-61  105-136 106-3   106-17
        106-126 107-3   107-17  107-70  108-3   108-17  108-87  109-3   109-17  109-129 110-3   110-17  110-39  110-74
        111-3   111-17  114-28  115-23  116-3   116-17  118-36  119-3   119-17  119-114 120-3   120-17  120-80  121-3
        121-17  121-70  122-3   122-17  122-44  123-3   123-17  123-100 124-3   124-17  124-46  125-3   125-17  129-27
        130-3   130-17  130-134 134-29  134-40  136-33  137-3   137-17  137-81  140-76  140-79  141-20  142-3   142-17
        142-113 142-156 143-20  144-3   144-17  144-92  144-108 144-108 144-108 144-108
UDA52     1-41#   1-54    1-63    1-80    4-16    4-46    4-102   6-59    8-86    33-10   41-17   62-22   63-22   71-12
         72-22   73-26   73-32   74-22   81-22   81-43   81-49   82-12   83-25   84-15   85-22   91-18   94-22   95-27
         95-50   96-31   96-64   98-22   98-45   99-22   100-22  100-59  100-90  101-22  101-45  102-17  104-22  104-50
        105-22  105-49  105-61  106-22  106-43  107-22  108-22  109-22  110-22  110-44  111-22  114-28  116-22  119-22
        120-22  121-22  121-85  122-22  123-22  124-22  125-22  130-22  130-134 134-29  134-40  137-22  137-81  140-76
        140-79  142-22  142-118 142-144 142-161 143-7   144-22  144-106 144-108
UDADM4    1-86#
ULOP    144-108 144-108#
UMASK   144-108 144-108# 144-108* 144-108*
UNADDR    6-61#   95-35   100-71  103-18  110-52  144-108
UNITS   144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108#
UNSSUC    7-44#   39-85
UP        74-34   74-87#
UPG       85-47   88-2#
UPT       85-45   86-2#   88-39
UREAD     5-14#
UTOTST    7-14#  144-108
UWRITE    5-15#
VALBIT  144-108 144-108#
VALID     5-39#   39-84   39-87   54-9    54-37
```

CROSS REFERENCE TABLE (CREF V04.00)

```
WAITSI    5-13#   99-12   102-8    142-140
WBLOCK   98-115   99-2#
WBUFLN    4-94#    4-95    95-13   144-108
WCHECK    9-54#   49-14    62-66    62-82    64-14    67-8    67-17    68-10   144-108
WCHKAL    9-56#   67-12   144-108
WCONT     4-69#
WONLY     9-47#   65-8    137-38   144-108
WREAL     4-70#   96-11
WRITBT   40-23#   56-61*
WRITE    96-84    98-24#
WSTOP     4-71#   96-13
XBNCYL    6-40#  130-108
XFERRT    6-5#
XOPLP0  109-67#  109-79
XOPLP1  109-71#  109-78
XOPLP2  109-81#  109-85
XREAD     5-5#   102-23   142-150
XWRITE    5-6#    99-28
ZEREDC   38-13    38-15#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
        144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108
        144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108
        144-108 144-108
BPL       1-83# 39-29   41-20   47-32   47-38   47-48   49-30   49-36   49-72   49-83   49-97   49-105  55-34   58-20
         58-30  76-15   78-12   84-6    85-52   98-120  99-41   106-57  109-107 110-62  112-9   119-49  123-97  126-25
        126-28  128-34  128-37  129-22  139-17  139-40  140-28  141-17  144-108 144-108 144-108
BR        1-83# 33-27   34-7    34-40   39-25   39-36   39-41   39-57   39-61   39-65   39-69   39-74   39-78   39-99
         41-17  41-22   44-30   44-40   47-43   47-51   49-37   49-54   49-80   49-111  49-114  49-124  50-52   52-22
         52-24  52-30   53-9    54-54   54-69   55-25   55-30   55-40   55-50   56-14   56-20   56-29   56-33   56-37
         56-57  57-21   58-27   58-34   62-64   62-81   62-96   62-102  62-122  63-39   63-41   63-45   63-49   66-11
         69-10  71-12   72-41   72-50   73-26   73-32   73-71   74-42   74-49   74-74   74-86   74-98   74-138  74-144
         74-148 76-13   77-16   78-16   80-41   80-52   80-58   80-75   81-43   81-49   82-12   82-21   83-25   84-8
         84-15  84-20   85-40   85-42   85-46   85-68   86-11   86-22   87-9    87-20   88-10   88-24   88-30   89-10
         89-22  91-18   91-27   93-11   94-39   95-19   95-37   96-15   96-52   96-76   96-85   96-87   98-45   98-59
         98-102 98-127  98-131  99-19   99-48   100-76  100-88  100-90  100-96  100-116 100-128 100-141 100-147 100-151
        100-156 100-161 100-165 100-168 100-179 100-181 101-45  101-67  101-72  102-14  103-23  104-82  104-95  104-108
        104-111 104-115 104-119 104-125 104-129 104-134 104-139 104-143 104-146 104-157 104-164 104-168 105-61  105-66
        105-72  105-81  105-97  105-101 105-103 105-122 105-128 105-130 105-133 106-54  106-67  106-90  106-108 106-110
        106-116 106-120 106-123 107-60  107-67  108-44  108-57  108-61  108-63  108-71  108-85  109-46  109-79  109-88
        109-113 109-118 109-127 110-56  110-71  111-71  111-82  111-89  112-15  112-20  113-31  113-43  113-49  114-28
        114-39  114-42  116-53  116-67  116-85  116-89  117-12  118-32  119-46  119-81  119-88  119-95  119-112 120-55
        120-63  120-66  120-77  121-62  122-41  123-41  123-84  124-37  124-43  125-45  125-55  129-24  130-49  130-85
        130-116 130-134 130-143 130-149 130-157 131-40  132-32  132-44  133-23  134-22  134-24  134-29  134-31  134-33
        134-40  134-49  134-54  134-68  134-70  137-41  137-81  137-90  137-96  137-100 138-17  138-19  138-28  138-37
        138-43  139-34  139-46  140-30  140-33  140-43  140-76  140-79  140-83  141-10  141-14  142-111 142-190 142-200
        142-231 144-43  144-49  144-68  144-74  144-89  144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108
        144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108
        144-108
CALL      1-83# 34-2    34-17   35-7    39-23   39-32   39-35   39-50   39-55   39-88   39-106  41-15   47-36   47-41
         47-53  49-13   49-43   49-77   49-79   49-118  49-121  49-125  49-128  50-46   50-51   52-13   53-6    54-15
         54-38  55-4    55-13   55-14   55-32   56-6    56-10   56-16   56-40   56-50   56-53   56-63   59-25   59-34
         59-43  59-53   59-57   62-115  62-118  65-12   66-12   67-14   68-14   72-44   73-16   73-22   73-29   73-56
         74-50  74-52   74-73   74-76   74-118  74-137  74-139  75-13   76-12   77-18   77-31   78-18   78-32   80-17
         80-27  80-29   80-47   81-34   81-39   81-46   83-21   83-22   84-10   84-11   85-39   85-41   85-45   85-47
         86-10  87-8    88-8    88-9    88-25   88-39   89-8    89-9    89-23   89-24   93-6    94-32   94-45   95-14
         95-17  95-40   96-21   98-39   98-60   98-113  98-115  99-17   99-46   99-51   100-89  100-115 100-144 100-146
        100-155 100-160 100-176 101-39  101-68  101-69  102-13  102-25  102-30  104-58  104-81  104-122 104-124 104-133
        104-138 104-154 105-60  105-83  105-92  106-70  106-86  106-89  107-32  108-47  108-67  109-93  111-81  111-83
        111-88  112-19  112-30  113-27  113-53  114-25  114-33  114-44  116-65  116-68  116-69  116-72  116-84  117-16
        118-10  119-41  119-47  119-63  119-70  119-91  119-96  119-111 120-40  120-51  120-54  121-57  123-34  123-61
        123-65  124-33  125-44  125-46  125-47  126-23  128-12  130-54  130-65  130-71  130-93  130-94  130-101 130-104
        130-120 131-3   131-20  132-11  132-24  134-13  134-28  134-39  134-57  137-47  137-54  137-61  137-67  138-14
        138-26  138-40  140-10  140-31  140-49  142-49  142-70  142-74  142-81  142-91  142-102 142-178 142-210 142-221
        142-230 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108 144-108
        144-108 144-108 144-108 144-108 144-108 144-108
CERROR   14-86# 34-6    34-8    39-76   39-83   56-13   56-19   56-28   56-32   56-34   56-56   56-64   100-169 104-147
        108-70  109-98  109-101 112-14  112-16  113-48  113-50  114-38  114-40  118-33  120-65  123-37  124-38  130-150
        134-47  134-48  134-65  134-66
DEVFTL   14-60# 52-20   56-25   99-15   99-44   102-11  116-59  133-22  134-21  134-23  134-30  134-53  134-67  134-69
        137-40  138-18  138-36  138-42  139-45  140-29  142-207 144-71  144-108 144-108
DFOVLY   11-3#
DIAG$$   10-5#
DMCODE    1-82#  1-86
DMEND     1-82#  144-113
DMODT     1-84#
DMOVLY    1-82#  1-86    144-109
```

UDAT4 DISK EXERCISER DMACR X04.01 23-AUG-82 12:02:53 SEQ 0851

TABLE OF CONTENTS

TABLE OF CONTENTS

UDAT4 DISK EXERCISER DMACR X04.01 23-AUG-82 12:02:53 SEQ 0853

TABLE OF CONTENTS

```
                                .TITLE  UDAT4   DISK EXERCISER


                        ;
                        ;
                        ; COPYRIGHT (C)  1981,1982
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;THIS PROGRAM SHALL BE ASSEMBLED WITH THE PROGRAM DMACRO
                        ;USING A COMMAND LINE SIMILAR TO:
                        ;
                        ;       UDAT4.BIN,UDAT4/C=$DMACRO,UDAT4
                        ;
                        ;       ONE, AND ONLY ONE OF THE FOLLOWING ASSEMBLY OPTIONS <<MUST>> BE
                        ;       SELECTED
                        ;
000001                  UDA50   =       1       ; SET TO 1 IF ASSEMBLY IS FOR UDA50
000000                  UDA52   =       0       ; SET TO 1 IF ASSEMBLY IS FOR UDA52
000000                  QDA     =       0       ; SET TO 1 IF ASSEMBLY IS FOR QDA
                        ;
                        ;
                        ;       OTHER OPTIONS
                        ;
000000                  DEBUG   =       0       ; SET TO 1 TO DEBUG TEST4
000000                  INCODT  =       0       ; MUST BE 0 FOR UDA50
                        ;       .ENABL  ABS
000000                          DMCODE  UDADM4,0,714,3,0,1000
```

```
                                .SBTTL  MODIFICATION CONTROL
000000                                  .REPT   0


                                        MODIFICATION CONTROL -- ALL CHANGES AFTER OCTOBER 29, 1981 WILL BE
                                        ENTERED IN THIS SECTION TO ALLOW THE HISTORY OF TEST 4 TO BE DOCUMENTED
                                        JULY , 1982
                                                THE "GET STATUS AFTER ERROR" CODE (IN ROUTINE CORECT)
                                                WAS CHANGED TO OVERCOME THE RA81 PROBLEM WHERE RECEIVER
                                                READY WAS NOT IMMEDIATELY ASSERTED AFTER AN ERROR (THE
                                                DRIVE WAS "PICKING UP THE PIECES" AFTER AN ERROR, AND
                                                ALL F'S USED TO BE DISPLAYED BECAUSE I WOULDN'T SEE
                                                RECEIVER READY WHEN I WANTED TO).  THIS REQUIRED MOVING
                                                THE "WAIT FOR CLOCKS" CODE, WHICH USED TO BE IN THE
                                                "GET STATUS AFTER ERROR" CODE (IN CORECT) TO THE MODULE
                                                GOINIT, AND REWRITING A SMALL AMOUNT OF CODE IN CORECT.

                                                ALL CODE ADDED FOR CONDITIONAL ASSEMBLY FOR THE
                                                UDA50/UDA52.
                                        .ENDR
```

```
                        .SBTTL  UDA DM PROGRAM PARAMETERS
                                .LIST MEB
                        ;
                        ;       EQUATES
                        ;
                        ;       HIGHEST USABLE LOCATION OF UDA MEMORY + 1
                        ;
007774                  HIMEM   =       7774            ; HIGH MEMORY+1
007774                  OVSTRT  =       7774            ; OVERLAY ADDRESS LOCATION
                        ;
                        ;
                        ;       OFFSETS FOR FORMAT TRACK TABLE
                        ;
000000                  FT.BUF  =       0.              ; BUFFER POINTER OFFSET
000001                  FT.HI   =       1.              ; HI ORDER HEADER OFFSET
000002                  FT.LOW  =       2.              ; LOW ORDER HEADER OFFSET
                        ;
                        ;
                        ;       OFFSETS FOR FORMAT TRACK BUFFER
                        ;
000000                  FB.DAT  =         0.            ; FIRST DATA WORD OFFSET
000400                  FB.EDC  =       256.            ; EDC WORD OFFSET
                        ;
                        ;
                        ;       OFFSETS FOR READ/WRITE I/O CHAIN TABLES
                        ;
000000                  RW.STAT =       0.              ; STATUS AND NEXT BUFFER POINTER OFFSET
000001                  RW.BUF  =       RW.STAT+1       ; POINTER TO DATA BUFFER
000002                  RW.LOW  =       RW.BUF+1        ; HI ORDER EXPECTED HEADER
000003                  RW.HI   =       RW.LOW+1        ; LOW ORDER EXPECTED HEADER
000004                  RW.CMD  =       RW.HI+1         ; SDI COMMAND AND HEAD ADDRESS
000005                  RW.SDI  =       RW.CMD+1        ; DUMMY SDI CONTROL BLOCK POINTER
000006                  RW.ANG  =       RW.SDI+1        ; THETA FROM INDEX
                        ;
                        ;       CONSTANTS FOR READ AND WRITE XFC'S
                        ;
040000                  BUFFLG  =       40000           ; BUFFER FULL OR EMPTY FLAG
010000                  ECCFLG  =       10000           ; ECC ERROR IN BUFFER BIT
100000                  EOC     =       100000          ; END OF CHAIN
100000                  FSTOP   =       100000          ; LAST ENTRY IN CHAIN FOR FORMAT
000000                  RCONT   =       0               ; READ CONTINUE
013400                  RREAL   =       13400           ; READ REAL TIME COMMAND
100000                  RSTOP   =       EOC             ; LAST ENTRY IN CHAIN FOR READ
040000                  WCONT   =       40000           ; WRITE CONTINUE
122400                  WREAL   =       122400          ; WRITE REAL TIME ECOMMAND
140000                  WSTOP   =       EOC+BUFFLG      ; LAST ENTRY IN CHAIN FOR WRITE
000400                  SEC512  =       256.            ; 256 WORDS IN A SECTOR IN 512 BYTE MODE
000440                  SEC576  =       288.            ; 288 WORDS IN A SECTOR IN 576 BYTE MODE
                        ;
                        ;
                        ;       HEADER CODES
                        ;
000000                  HD.LBN  =       000000          ; GOOD LBN
060000                  HD.RBN  =       060000          ; GOOD RBN, PERHAPS UNUSED
030000                  HD.REV  =       030000          ; REVECTORED LBN
110000                  HD.BAD  =       110000          ; BAD BLOCK
```

```
      82          050000                HD.PRV  =       050000          ; PRIMARY REVECTORED BLOCK
      83          120000                HD.XBN  =       120000          ; XBN BLOCK
      84          140000                HD.DBN  =       140000          ; DBN BLOCK
      85
      86                                ;       OFFSETS FOR DATA BUFFERS
      87
      88          000000                BF.DAT  =         0.            ; DATA
      89          000400                BF.EDC  =       256.            ; ERROR DETECTION CODE
      90          000401                BF.ECC  =       257.            ; LAST 12 ECC RESIDUES
      91
      92                                ;       BUFFER AND READ/WRITE CHAIN LINK SIZES
      93
      94          000401                WBUFLN  =       257.            ; WRITE BUFFER SIZE
      95          000415                RBUFLN  =       WBUFLN+12.      ; READ BUFFER SIZE
      99          000007                LINKLN  =       7.              ; LINK SIZE
```

```
                        ;       XFC DEFINITION EQUATES
                        ;
000000                  BREAK   =       0.              ; BREAKPOINT XFC CODE
000001                  FORMAT  =       1.              ; FORMAT TRACK XFC CODE
000002                  XREAD   =       2.              ; READ N SECTORS XFC CODE
000003                  XWRITE  =       3.              ; WRITE N SECTORS XFC CODE
000004                  SEND    =       4.              ; SEND SDI COMMAND XFC CODE
000005                  RCV     =       5.              ; RECEIVE SDI MESSAGE XFC CODE
000006                  COMPARE =       6.              ; COMPARE DATA PATTERN TO BUFFER
000007                  STATUS  =       7.              ; RETURN DRIVE STATUS XFC CODE
000010                  ECHO    =       8.              ; ECHO DATA TO DRIVE XFC CODE
000011                  DINIT   =       9.              ; DRIVE INITIALIZE XFC CODE
000012                  WAITSI  =       10.             ; WAIT FOR SECTOR OR INDEX PULSE
000013                  UREAD   =       11.             ; READ UNIBUS MEMORY XFC CODE
000014                  UWRITE  =       12.             ; WRITE UNIBUS MEMORY XFC CODE
000015                  ECC     =       13.             ; DO ECC ON BUFFER XFC CODE
000016                  MRD     =       14.             ; SEND BUFFER TO MAINTENANCE READ COMMAND
000017                  MWR     =       15.             ; GET BUFFER FROM MAINTENANCE WRITE COMMAND
000020                  CVT     =       16.             ; CONVERT TO PHYSICAL ADDRESS XFC CODE
000021                  EXIT    =       17.             ; TERMINATE DM PROGRAM XFC CODE
                        ;
                        ;       MEDIA TYPES
                        ;
                        ;       THIS WORD IS FOUND AS THE FIRST WORD OF XBN 0 (OR ANY OF ITS COPIES)
                        ;
126736                  MOD512  =       126736          ; 512 BYTE MODE
074161                  MOD576  =       074161          ; 576 BYTE MODE
                        ;
                        ;       GET STATUS OFFSETS
                        ;
000000                  ST.UNT  =       0.              ; UNIT NUMBER
000000                  ST.MSK  =       0.              ; SUBUNIT MASK
000001                  ST.REQ  =       1.              ; REQUEST BYTE
000001                  ST.MOD  =       1.              ; MODE BYTE
000002                  ST.ERR  =       2.              ; ERROR BYTE
000002                  ST.CON  =       2.              ; CONTROLLER BYTE
000002                  ST.C    =       2.              ; C BITS
000003                  ST.RTY  =       3.              ; RETRY COUNT/FAILURE CODE
100000                  VALID   =       100000          ; USED IN ERROR POSITION TO MARK VALID STATUS
                                                        ;  IN INPUT BUFFER
                        ;
                        ;       STATUS BIT DEFINITIONS
                        ;
000200                  ST.OA   =       200             ; ONLINE TO ANOTHER (SET IF DRIVE UNAVAILABLE)
000100                  ST.RR   =       100             ; READJUSTMENT BIT (SET IF RECALIBRATION REQUIRED)
000040                  ST.DR   =       40              ; DIAGNOSTIC REQUEST (SET IF DIAGNOSTIC REQUESTED)
000020                  ST.SR   =       20              ; SPINDLE READY (SET IF SPINDLE READY)
000010                  ST.EL   =       10              ; LOGGABLE INFO IN EXTENDED STATUS
000002                  ST.PS   =       2               ; PORT SWITCH (SET IF PORT SWITCH IN)
000001                  ST.RU   =       1               ; RUN/STOP SWITCH (SET IF RUN/STOP SWITCH IN)
000200                  ST.DE   =       200             ; DRIVE ERROR
000100                  ST.RE   =       100             ; RETRIABLE ERROR (SET IF RETRIABLE ERROR OCCURRED)
000040                  ST.PE   =       40              ; PROTOCOL ERROR (SET IF PROTOCOL ERROR OCCURRED)
000020                  ST.DF   =       20              ; INITIALIZATION FAILURE (SET IF INIT FAILED)
000010                  ST.WE   =       10              ; WRITE ENABLE (SET IF WRT ATTEMPTED ON PROT DISK)
002000                  ST.FO   =       2000            ; FORMATTING (SET IF FORMATTING ENABLED)
001000                  ST.DB   =       1000            ; DIAGNOSTIC CYLS (SET IF DIAG CYL ACCESS ENABLED)
```

```
     58          000400                    ST.S7   =       400             ; SECTOR SIZE (SET FOR 576 BYTE SECTORS)
```

```
                                ;       GET COMMON CHARACTERISTICS OFFSETS
                                ;
000000                          SHRTTO  =       0.              ; SHORT TIMEOUT <3:0>
000000                          SDIVER  =       0.              ; SDI VERSION <7:4>
000000                          XFERRT  =       0.              ; TRANSFER RATE <15:0>
000001                          LONGTO  =       1.              ; LONG TIMEOUT <3:0>
000001                          RETS    =       1.              ; RETRIES <7:4>
000001                          RCTCPS  =       1.              ; F/RCT COPIES <11:8>
000001                          SS      =       1.              ; SECTOR SIZE <15:15>
000002                          ERLEV   =       2.              ; ERROR RETRY LEVELS <7:0>
000002                          ECCRSH  =       2.              ; ECC THRESHOLD <15:8>
000003                          MICREV  =       3.              ; MICROCODE REVISION NUMBER <7:0>
000003                          HRDREV  =       3.              ; HARDWARE REVISION NUMBER <15:8>
000004                          DRVID   =       4.              ; UNIQUE DRIVE ID <47:0>
000007                          DRTYPE  =       7.              ; DRIVE TYPE IDENTIFIER <7:0>
000007                          REVS    =       7.              ; REVS/SECOND <15:8>
                                ;
                                ;       GET SUBUNIT CHARACTERISTICS OFFSETS
                                ;
                                ;THESE OFFSETS ARE CURRENTLY GIVEN AS FOLLOWING THE COMMON CHARACTERISTICS
                                ;
000000                          LBNCYL  =       0.              ; NUMBER OF CYLINDERS IN LBN AREA <31:0>
000001                          HICYL   =       1.              ; HI ORDER CYLINDER BITS <15:12>
000002                          GRPCYL  =       2.              ; GROUPS PER CYLINDER <7:0>
000002                          HILBN   =       2.              ; HI STARTING LBN <11:8>
000002                          HIXBN   =       2.              ; HI STARTING XBN <15:12>
000003                          TRKGRP  =       3.              ; TRACKS PER GROUP <7:0>
000003                          HIRBN   =       3.              ; HI STARTING RBN <11:8>
000003                          HIDBN   =       3.              ; HI STARTING DBN <15:12>
000004                          RBNTRK  =       4.              ; RBNS PER TRACK <6:0>
000004                          RM      =       4.              ; REMOVABLE MEDIA <7:7> 1=REMOVEABLE
000005                          DATPRE  =       5.              ; DATA PREAMBLE SIZE IN WORDS <7:0>
000005                          HDRPRE  =       5.              ; HEADER PREAMBLE SIZE IN WORDS <15:8>
000006                          MEDTYP  =       6.              ; MEDIA TYPE <31:0>
000010                          FCTSIZ  =       8.              ; FCT COPY SIZE <15:0>
000011                          LBNTRK  =       9.              ; LBNS PER TRACK <7:0>
000011                          GRPOFF  =       9.              ; GROUP OFFSET (SECTORS) <15:8>
000012                          LBNHST  =       10.             ; LBNS IN HOST AREA <31:0>
000014                          RCTCSZ  =       12.             ; RCT COPY SIZE <15:0>
000021                          XBNCYL  =       17.             ; CYLS IN XBN AREA <15:0>
000022                          DBNCYL  =       18.             ; CYLS IN DBN AREA <15:8>
                                ;
                                ;
                                ;       BIT MASK DEFINITIONS
                                ;
                                ;
177400                          HIBYTE  =       177400          ; HIGH BYTE MASK
000377                          LOBYTE  =       000377          ; LOW BYTE MASK
007777                          HBHINB  =       7777            ; HI BYTE, HI NIBBLE MASK
170377                          HBLONB  =       170377          ; HI BYTE, LO NIBBLE MASK
177417                          LBHINB  =       177417          ; LO BYTE, HI NIBBLE MASK
177760                          LBLONB  =       177760          ; LO BYTE, LO NIBBLE MASK
170000                          UNADDR  =       170000          ; UNUSED ADDRESSING BITS
                                ;
```

```
                                ;MAINTANENCE READ/WRITE REQUEST NUMBERS
                                ;
060000                          DUPPKT  =       060000          ; DUP SPECIAL PACKET NUMBER
060000                          T1MSI7  =       0.+DUPPKT       ; GET FREE MEMORY PARAMETERS
060001                          T2DLL   =       1.+DUPPKT       ; DOWNLINE LOAD DRIVE DIAGNOSTIC
060002                          T2CMD   =       2.+DUPPKT       ; MANUAL INTERVENTION TEST 2 PROTOCOL
060003                          T4MPRM  =       3.+DUPPKT       ; GET MASTER PARAMETERS FROM SW QUESTIONS
060004                          T4UPRM  =       4.+DUPPKT       ; GET UNIT PARAMETERS FROM HW QUESTIONS
060005                          T4BB1   =       5.+DUPPKT       ; GET BAD BLOCKS (1 THRU 14)
060006                          T4BB2   =       6.+DUPPKT       ; GET REST OF BAD BLOCKS (15 AND 16)
060007                          T4SOFT  =       7.+DUPPKT       ; ADD TO SOFT ERROR AND ECC COUNT
060010                          T4SEEK  =       8.+DUPPKT       ; ADD 1 TO SEEK COUNT
060011                          T4MXFR  =       9.+DUPPKT       ; ADD TO MEGABITS READ AND WRITTEN
060012                          UTOTST  =       10.+DUPPKT      ; GET UNITS TO TEST
060013                          ERRMES  =       11.+DUPPKT      ; PRINT ERROR MESSAGE
060014                          ERRMC   =       12.+DUPPKT      ; TEST 4 ERROR REPORTING
060015                          MESSAG  =       13.+DUPPKT      ; INFORMATION MESSAGE
060016                          DONE    =       14.+DUPPKT      ; MARK DM PROGRAM AS NO LONGER RUNNING
                                ;
                                ;       STATE BIT DEFINITIIONS
                                ;
000001                          RCVRDY  =       1               ; RECIEVER READY 1 = READY
000002                          ATTN    =       2               ; ATTENTION BIT FOR RETURN DRIVE SIGNALS XFC
000004                          DCLOCK  =       4               ; CONTROLLER STATE READY -- THERE ARE CLOCKS
                                                                ; COMING FROM THE DRIVE
000100                          AVAIL   =       100             ; AVAILABLE 1 = AVAILABLE
000400                          RCVERR  =       400             ; RECEIVE ERROR -- PARITY ERROR
100000                          RWRDY   =       100000          ; IF SET, UDA IS ABLE TO READ AND/OR WRITE TO DRIVE
                                ;
                                ;       SDI COMMANDS AND RESPONSES
                                ;
000204                          DISCON  =       204             ; DISCONNECT DRIVE
000006                          ERECOV  =       6               ; ERROR RECOVERY
000201                          CHGMOD  =       201             ; CHANGE MODE
000213                          DRVONL  =       213             ; DRIVE ONLINE
000014                          DRVRUN  =       14              ; DRIVE RUN
000005                          DRVCLR  =       5               ; DRIVE CLEAR OPCODE
000207                          GETCHR  =       207             ; GET CHARACTERISTICS
000210                          GETSUB  =       210             ; GET SUBUNIT CHARACTERISTICS
000011                          GETSTA  =       11              ; GET STATUS
000216                          IRECLB  =       216             ; RECALIBRATE
000012                          INSEEK  =       12              ; INITIATE SEEK
000176                          COMPLT  =       176             ; SUCCESSFUL COMPLETION
000175                          UNSSUC  =       175             ; UNSUCCESSFUL COMPLETION
000170                          CHRRES  =       170             ; GET CHARACTERISTICS RESPONSE
000167                          SBCRES  =       167             ; GET SUBUNIT CHARACTERISTICS RESPONSE
000366                          STSRES  =       366             ; GET STATUS RESPONSE
000350                          ECHOC   =       350             ; DIAGNOSTIC ECHO COMMAND AND RESPONSE
                                ;
                                ;       SDI LEVEL 2 COMMAND OFFSETS
                                ;
000000                          L2.OPC  =       0               ; SDI LEVEL 2 OP CODE
000001                          L2.SLN  =       L2.OPC+1        ; SDI LEVEL 2 SEND LENGTH IN BYTES
000002                          L2.RLN  =       L2.SLN+1        ; SDI LEVEL 2 RECEIVE LENGTH IN WORDS
000003                          L2.ERS  =       L2.RLN+1        ; SDI LEVEL 2 EXPECTED RESPONSE
000004                          L2.EOF  =       L2.ERS+1        ; SDI LEVEL 2 ERROR OFFSET INTO UNIT PARAMETERS
                                ;
```

```
58                                      ;       ERROR CODES
59                                      ;
60          000000                      FTLSYS  =       0                       ; SYSTEM FATAL ERROR
61          040000                      FTLDEV  =       040000                  ; DEVICE FATAL
62          100000                      ERHARD  =       100000                  ; HARD ERROR
63          140000                      ERSOFT  =       140000                  ; SOFT ERROR
64          040000                      C2HARD  =       <ERHARD&^CERSOFT>!<ERSOFT&^CERHARD>     ; CHANGE SOFT TO HARD ERROR
65          100000                      C2DFTL  =       <FTLDEV&^CERSOFT>!<ERSOFT&^CFTLDEV>     ; CHANGE SOFT TO DEVICE FATAL
```

```
                        .SBTTL  TEST 4 SPECIFIC INFORMATION
                        ;
                        ;       TEST 4 SPECIFIC INFORMATION
                        ;
                        ;
                        ;
                        ;       UNIT PARAMETER OFFSETS
                        ;
                        ;
000000                  U.NEXT  =       0.              ; POINTER TO NEXT UNIT (RING LINKED LIST)
000001                  U.SUBP  =       U.NEXT+1        ; 4 WORDS OF SUBUNIT PARAMETER POINTERS
000005                  U.TIMH  =       U.SUBP+4        ; HI ORDER TIMEOUT FOR SEEK
000006                  U.TIML  =       U.TIMH+1        ; LO ORDER TIMEOUT FOR SEEK
000007                  U.SRTY  =       U.TIML+1        ; ISSUE SEEK TIMEOUT COUNTER
000010                  U.RRTY  =       U.SRTY+1        ; READ RETRIES
000011                  U.MSTO  =       U.RRTY+1        ; MASTER SEEK TIMEOUT
000012                  U.RWTO  =       U.MSTO+1        ; READ/WRITE TIMEOUT AREA
000013                  U.NFUN  =       U.RWTO+1        ; NEXT FUNCTION ADDRESS (FOR DEFERRED CALLS)
000014                  U.PAT   =       U.NFUN+1        ; PATTERN NUMBER TO WRITE
000015                  U.CCNT  =       U.PAT+1         ; CURRENT COUNT OF T/G LOOPS
000017                  U.PCTG  =       U.CCNT+2        ; POINTER TO CURRENT TRACK OR GROUP
000020                  U.CTRK  =       U.PCTG+1        ; TRACK COUNT FOR GROUP OPERATIONS
000021                  U.NSEC  =       U.CTRK+1        ; NUMBER OF SECTORS R/W THIS TRY
000022                  U.MSEC  =       U.NSEC+1        ; NUMBER OF SECTORS TO BE R/W
000023                  U.TSEC  =       U.MSEC+1        ; NUMBER OF SECTORS TO BE R/W THIS OP
000024                  U.CSEC  =       U.TSEC+1        ; COUNT OF SECTORS R/W SO FAR
000025                  U.MASK  =       U.CSEC+1        ; UNIT MASK FOR XFC CALLS (0001 - 1000)
000026                  U.WRIT  =       U.MASK+1        ; WRITE PROTECTION STATUS
000027                  U.ELEV  =       U.WRIT+1        ; CURRENT ERROR RECOVERY LEVEL
000030                  U.RTRY  =       U.ELEV+1        ; MAXIMUM NUMBER OF READ RETRIES
000031                  U.MLEV  =       U.RTRY+1        ; MAXIMUM NUMBER OF ERROR RECOVERY LEVELS
000032                  U.ECCT  =       U.MLEV+1        ; ECC THRESHOLD
000033                  U.SDIS  =       U.ECCT+1        ; SDI SHORT TIMEOUT
000034                  U.SDIL  =       U.SDIS+1        ; SDI LONG TIMEOUT
000035                  U.SDI2  =       U.SDIL+1        ; LEVEL 2 ERROR COUNTS
000045                  U.WPRT  =       U.SDI2+NUML2S   ; MASK TO WRITE PROTECT READ-ONLY DRIVES
000046                  U.PARM  =       U.WPRT+1        ; UNIT PARAMETER WORD
000047                  U.RCOV  =       U.PARM+1        ; RECOVERY PARAMETERS
000050                  U.SUBU  =       U.RCOV+1        ; SUBUNIT OFFSET (0 - 3)
000051                  U.MBN   =       U.SUBU+1        ; MASTER L/DBN
000053                  U.CBN   =       U.MBN+2         ; CURRENT L/DBN FOR START OF CHAIN
000055                  U.RBN   =       U.CBN+2         ; RBN TO BE READ/WRITTEN IF LBN REVECTORED
000057                  U.COPY  =       U.RBN+2         ; NUMBER OF RCT COPIES ON EACH SUBUNIT
000060                  U.CCOP  =       U.COPY+1        ; CURRENT RCT COPY THAT WE'RE WORKING ON
000061                  U.RWER  =       U.CCOP+1        ; ERROR (IF ANY) ON THE LAST R/W
000062                  U.RVER  =       U.RWER+1        ; ERROR THAT OCCURRED BEFOR THE REVECTOR OPERATION
000063                  U.UNUM  =       U.RVER+1        ; STARTING SUBUNIT NUMBER
000064                  U.CCYL  =       U.UNUM+1        ; CURRENT CYLINDER
000066                  U.CGRP  =       U.CCYL+2        ; CURRENT GROUP
000067                  U.LCYL  =       U.CGRP+1        ; LAST CYLINDER SEEKED TO
000071                  U.LGRP  =       U.LCYL+2        ; LAST GROUP SEEKED TO
                        ;
                        ;       SUBUNIT PARAMETER OFFSET
                        ;
000000                  S.PARM  =       0.              ; SUBUNIT PARAMETER WORD
000001                  S.SEEK  =       S.PARM+1        ; COUNT OF NUMBER OF SEEKS
000002                  S.SDCL  =       S.SEEK+1        ; STARTING DIAGNOSTIC CYLINDER
```

```
     58          000004                  S.PAT   =       S.SDCL+2        ; PATTERN TO USE FOR WRITES
     59          000005                  S.LETR  =       S.PAT+1         ; L OR D LETTER TO MAKE L/DBN
     60          000006                  S.TRKL  =       S.LETR+1        ; NUMBER OF SECTORS IN ONE TRACK
     61          000007                  S.SCHR  =       S.TRKL+1        ; POINTER TO SUBUNIT CHARACTERISTICS
     62          000010                  S.MEGR  =       S.SCHR+1        ; SECTORS READ (UP TO 245)
     63          000011                  S.MEGW  =       S.MEGR+1        ; SECTORS WRITTEN (UP TO 245)
     64          000012                  S.BADP  =       S.MEGW+1        ; POINTER TO BAD BLOCK AREA
     65          000013                  S.BESS  =       S.BADP+1        ; START OF BEGIN/END SETS
     66                                  ;
     67                                  ;       IF TRACK/GROUP LIMITS ARE GIVEN, THE SUBUNIT PARAMETERS HAVE THE
     68                                  ;       FOLLOWING FIELDS ADDED TO THEM
     69                                  ;
     70          000013                  S.MCNT  =       S.BESS          ; MAXIMUM TRACK/GROUP COUNT
     71          000015                  S.TGOF  =       S.MCNT+2        ; ORIGINAL TRACK/GROUP OFFSET
     72          000017                  S.TGSS  =       S.TGOF+2        ; START OF TRACK/GROUP SETS
     73                                  ;
     74                                  ;       CONSTANTS
     75                                  ;
     76          000377                  SCTWRD  =       255.            ; NUMBER OF WORDS IN SECTOR TO FILL
     77          000105                  INTEDC  =       69.             ; INITIAL EDC VALUE
     78          000072                  TLEN.U  =       U.LGRP+1        ; UNIT PARAMETER LENGTH
     79          007702                  FIRSTU  =       HIMEM-TLEN.U    ; LOCATION OF FIRST UNIT PARAMETER BLOCK
     83          177774                  MAXMSK  =       177774          ; TO DETERMINE MAXIMUM SECTORS TO READ/WRITE
     92                                                                  ; MAXIMUM SECTOR MASK IS THE 2'S COMPLEMENT
     93                                                                  ; OF THE MAXIMUM NUMBER OF SECTORS TO WRITE
     94          001750                  MAXSND  =       1000.           ; MAXIMUM TIMES TO SEND A COMMAND TO AN OFFLINE DRIVE
```

```
                        ;
                        ;       DUMMY SDI CONTROL BLOCK OFFSETS
                        ;
000001                  D.LIMT  =       1               ; DUMMY SDI SEARCH LIMIT
000002                  D.SCHR  =       2               ; DUMMY POINTER TO SUBUNIT CHAR-5
                        ;
                        ;       MASTER PARAMETER BITS (M.PARM)
                        ;
100000                  IWIPRG  =       100000          ; INITIAL WRITE IN PROGRESS
000002                  DIE     =       000002          ; ERRORS DURING INITIALIZATION
000001                  INTINP  =       000001          ; INITIAL SETUP IN PROGRESS
                        ;
                        ;       UNIT PARAMETER BITS (U.PARM(R5))
                        ;
100000                  DROP    =       100000          ; DROP BIT (SET IF UNIT OR SUBUNIT DROPPED)
040000                  INITW   =       40000           ; INITIAL WRITE (SET IF INITIAL WRITE IN PROG)
010000                  DIREC   =       10000           ; DIRECTION (SET IF SEQUENTIAL ACCESSES DECREASING)
004000                  NEWSUB  =       4000            ; SET IF SEQUENTIAL SEEKS MOVED TO NEW SUBUNIT
002000                  SEKINP  =       2000            ; SEEK IN PROGRESS - SET IF TRUE
001000                  FTIME   =       1000            ; FIRST TIME FLAG - SET FOR INIT CODE
000400                  REVEC   =       400             ; REVECTOR BIT (SET IF BLOCK REVECTORED)
000200                  RBNBN   =       200             ; SEE IF WORKING ON RBN
000100                  REDWRT  =       100             ; REDWRT (READ OR WRITE IN PROGRESS SET IF WRITE)
                        :               10              ; RESERVED (SEE SUBUNIT PARAMETER BITS)
                        :               2               ; RESERVED (SEE SUBUNIT PARAMETER BITS)
                        ;
                        ;       ERROR RECOVERY PARAMETERS (U.RCOV)
                        ;
100000                  ERMASK  =       100000          ; IF 1, ERROR RECOVERY IS DISABLED
040000                  DRINIT  =       40000           ; IF 1, INDICATES THAT DRIVE WAS INITIALIZED
020000                  LEVUSD  =       20000           ; IF 1, RE-ISSUE THIS LEVEL OF ERROR RECOVERY
010000                  NXTLEV  =       10000           ; IF 1, ISSUE NEW LEVEL OF ERROR RECOVERY
004000                  SEKREQ  =       4000            ; IF 1, SEEK IS REQUIRED FOR ERROR RECOVERY
002000                  RCBREQ  =       2000            ; IF 1, RECALIBRATE IS REQUIRED FOR ERROR RECOVERY
001000                  RETRY   =       1000            ; IF 1, JUST RETRY THE SAME MODULE
                        ;
                        ;       SUBUNIT PARAMETER BITS (S.PARM(R4))
                        ;
                        ;               100000          ; RESERVED (DROP BIT)
                        ;               40000           ; RESERVED (INITIAL WRITE)
020000                  DCYLS   =       20000           ; DIAGNOSTIC CYLINDER FLAG (SET IF DBNS)
010000                  ECCCHK  =       10000           ; 1 IF ECC CORRECTION ALLOWED   MASTER BITS
004000                  RONLY   =       4000            ; READ ONLY (SET IF READ ONLY)
002000                  WONLY   =       2000            ; WRITE ONLY (SET IF WRITE ONLY)
001000                  RTRIES  =       1000            ; 1 IF RETRIES ALLOWED
000200                  ONLYCL  =       200             ; SET IF ONLY CYLINDERS SPECIFIED
                                                        ; USED DURING SETUP ONLY
000100                  SEQSEK  =       100             ; SEQUENTIAL SEEK (START UP TESTING ONLY)
000040                  BEUSED  =       40              ; BEGIN/END SETS (USED IF SET)
000020                  TRACKS  =       20              ; TRACKS OR GROUPS (TRACK IF SET)
000010                  WCHECK  =       10              ; WRITE CHECK BIT (IF SET, WRITE CHECK WILL BE DONE)
                        ;                       WCHECK IS ALSO USED FOR THE UNIT PARAMETERS
000004                  WCHKAL  =       4               ; SET IF WRITE CHECK ALWAYS TO BE DONE
000002                  DATCMP  =       2               ; DATA COMPARE (SET IF DATA COMPARE TO BE DONE)
```

```
     58                                ;
     59          000001                DCMPAL  =       1       DATCMP IS ALSO USED FOR THE UNIT PARAMETERS
                                                                    ; SET IF DATA COMPARE ALWAYS TO BE DONE
```

MACRO DEFINITIONS

```
                        .SBTTL  MACRO DEFINITIONS
                        ;
                        ;       DIAGNOSTIC MACRO FOR TEST4 OVERLAYS
                        ;
                                .MACRO  DIAG$$
                                TST     $$DIAG+$DIAG$
                                BEQ     .+6
                                MOV     #60000,R0
                                MOV     R0,@$$DIAG+$DIAG$
                                .NLIST
                                .IF     NE,UDA50
                                .LIST
                                BR      .+1
                                .NLIST
                                .ENDC
                                .IF     NE,UDA52
                                .LIST
                                .ERROR  ; NOT CHECKED OUT YET
                                BR      .+2
                                .NLIST
                                .ENDC
                                .LIST
                        $DIAG$  =       $DIAG$ + 1
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;       OVERLAY INFORMATION MACRO
                        ;
                                .MACRO  DFOVLY  LABL$,ONAME,SAREA,EAREA
                                .WORD   <SAREA-PART0>/2
                                .IF     NB,EAREA
                                .WORD   <EAREA-SAREA>/2
                                .IFF
                                .WORD   0
                                .ENDC
                                .WORD   0
                                .WORD   OVL.'ONAME'*4
                                .IF     NE,OCNT$-LABL$
                                .ERROR  ; OVERLAY NUMBER AND POSITION IN TABLE DO NOT MATCH
                                .ENDC
                        OCNT$   =       OCNT$+1
                                .ENDM
                        ;
                        ;       MESSAGE CONTROL TABLE MACRO
                        ;
                                .MACRO  MSG CMDBUF,CMDSZ,RPLSZ,SUCCOM
                                .WORD   CMDBUF                  ; ADDRESS OF COMMAND
                                .WORD   CMDSZ                   ; SIZE OF COMMAND IN BYTES
                                .WORD   RPLSZ                   ; SIZE OF REPLY IN WORDS
                                .WORD   SUCCOM                  ; SUCCESSFUL COMPLETION CODE
                                .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
                        SDI$    =       SDI$+1
                                .ENDM
```

MACRO DEFINITIONS

```
      1                                 .MACRO  BCS     LAB..
      2                                 .NLIST
      3                                 .IF     NE,UDA50
      4                                 .LIST
      5                                                         BCC     .+2
      6                                 .NLIST
      7                                 .ENDC
      8                                 .IF     NE,UDA52
      9                                 .LIST
     10                                                         BCC     .+3
     11                                 .NLIST
     12                                 .ENDC
     13                                 .LIST
     14                                                         BR      LAB..
     15                                 .ENDM
```

MACRO DEFINITIONS

```
                        ;       PUSH REGISTER MACRO
                        ;
                                .MACRO PUSH R9
                                .IRP X,<R9>
                                MOV X,-(SP)
                                .ENDR
                                .ENDM
                        ;
                        ;       POP REGISTER MACRO
                        ;
                                .MACRO POP R9
                                .IRP X,<R9>
                                MOV (SP)+,X
                                .ENDR
                                .ENDM
```

```
                        ;
                        ;       PRIMARY ERROR REPORTING (TEST 4)
                        ;

                        .MACRO  SOFTER  NUM,ARGS
                        ERROR   ERSOFT,NUM,<ARGS>
                                                                MOV     #ERRMES,HRQ.RQ
                        .ENDM

                        .MACRO  REPSFT  SFTFLG,ECCFG,SEKFLG,COMFLG
                        .IF     NB,SFTFLG
                        .IF     IDN,SFTFLG,SOFT
                                                                MOV     #1,HRQ.02
                        .IFF
                        .ERROR  ; NON SOFT ERROR FLAG IN SOFT ERROR POSITION
                        .ENDC
                        .IFF
                                                                CLR     HRQ.02
                        .ENDC
                        .IF     NB,ECCFG
                        .IF     IDN,ECCFG,ECC
                                                                MOV     #1,HRQ.03
                        .IFF
                        .ERROR  ; NON ECC ERROR FLAG IN ECC ERROR POSITION
                        .ENDC
                        .IFF
                                                                CLR     HRQ.03
                        .ENDC
                        .IF     NB,SEKFLG
                        .IF     IDN,SEKFLG,SEEK
                                                                MOV     #1,HRQ.04
                        .IFF
                        .ERROR  ; NON SEEK ERROR FLAG IN SEEK ERROR POSITION
                        .ENDC
                        .IFF
                                                                CLR     HRQ.04
                        .ENDC
                ;       .IF     NB,COMFLG
                ;       .IF     IDN,COMFLG,COMM
                ;                                               MOV     #1,HRQ.05
                ;       .IFF
                ;       .ERROR  ; NON COMMUMICATION ERROR FLAG IN COMMUNICATION ERROR POSITION
                ;       .ENDC
                ;       .IFF
                ;                                               CLR     HRQ.05
                ;       .ENDC
                                                                PUSH    R0
                                                                MOV     U.UNUM(R5),R0
                                                                ADD     U.SUBU(R5),R0
                                                                MOV     R0,HRQ.01
                                                                MOV     #T4SOFT,R0
                                                                CALL    HOSTRQ
                                                                POP     R0
                        .ENDM

                        .MACRO  HARDER  NUM,ARGS
                        ERROR   ERHARD,NUM,<ARGS>
                                                                MOV     #ERRMC,HRQ.RQ
```

```
                        .ENDM

                        .MACRO  DEVFTL  NUM,ARGS
                        ERROR   FTLDEV,NUM,<ARGS>
                                                        MOV     #ERRMC,HRQ.RQ
                        .ENDM

                        .MACRO  SYSFTL  NUM,ARGS
                        ERROR   FTLSYS,NUM,<ARGS>
                                                        MOV     #ERRMC,HRQ.RQ
                        .ENDM

                        .MACRO  ERROR   TYPE,NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       5
                        MOVMSG  #ER'NUM,4
                        .IF     NB,<ARGS>
                         IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC
                                                        MOV     #NUM!TYPE+4000.,R2
                                                        MOV     R2,HRQ.02
                                                        MOV     #.,HRQ.01

                        .RADIX
                        .ENDM

                        .MACRO  CERROR  STNUM,ARGS
                        .RADIX  10
                NUMPTR  =       STNUM
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .RADIX
                        .ENDM

                        .MACRO  ERRORC  ARGS
                        .RADIX  10
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .RADIX
                        .ENDM

                        .MACRO  MOVMSG  ARG,INDX

                        .IF     LT,INDX-10
                                                        MOV     ARG,HRQ.0'INDX
                        .IFF
                                                        MOV     ARG,HRQ.'INDX
                        .ENDC
                        .ENDM

                        .MACRO  ENDERR  POS
```

```
                        .RADIX  10
                        .IF     NB,POS
                        NDERR   POS
                        .IFF
                        NDERR   \NUMPTR
                        .ENDC
                        .RADIX
                        .ENDM

                        .MACRO  NDERR   POS
                        .IF     NE,POS
                        MOVMSG  #SER22,POS
                                                                MOV     #POS+1,ERRPOS   ; SET THE POSITION
                        .IF     LT,26.-POS
                        .ERROR  ; STATUS AT END OF ERROR MESSAGE WILL OVERFLOW HOST COMMUNICATION BUFFER
                        .ENDC
                        .IFF
                                                                CLR     ERRPOS          ; CLEAR THE POSITION
                        .ENDC
                        .ENDM
                ;
                ;       MESSAGE REPORTING MACRO
                ;
                        .MACRO  MSSG    NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       3
                        MOVMSG  #MS'NUM,2
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC
                                                                PUSH    R0
                                                                MOV     U.UNUM(R5),HRQ.01
                                                                ADD     U.SUBU(R5),HRQ.01
                                                                MOV     #MESSAG,R0
                                                                CALL    HOSTRQ
                                                                POP     R0
                        .RADIX
                        .ENDM
                ;
                ;
```

MACRO DEFINITIONS

```
                        ;ASSUME MACRO
                                .MACRO  ASSUME  P1,P2
                                .IF     NE,P1-P2
                                .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                .ENDC
                                .ENDM
                        ;
                        ;       XOR THE CONTENTS OF TWO REGISTERS
                        ;
                                .MACRO  RXOR    REG1,REG2
                                MOV     REG2,-(SP)      ; SAVE REGISTER REG2
                                BIC     REG1,REG2               ; CLEAR COORESPONDING BITS IN REG2
                                BIC     (SP)+,REG1      ; CLEAR COORESPONDING BITS IN REG1
                                BIS     REG1,REG2               ; OR WHAT'S LEFT
                                .ENDM
                        ;
                        ;       CONVERT .BLKW CALLS TO ACTUAL WORDS GENERATED
                        ;
                                .MACRO  .BLKW   COUNT
                                .NLIST
                                .REPT   COUNT
                                .WORD   0
                                .ENDR
                                .LIST
                                .ENDM
                        ;
                        ;       MASSIVE INLINE CODE MACROS NOT PRINTED
                        ;
```

```
                                .SBTTL  START OF TEST CODE
                                ;THE FOLLOWING IS FOR DEBUG PURPOSES ONLY. CAN BE NOP OR BREAKPOINT.

 000714  114007                         CLR     R0              ; CHANGE TO BREAKPOINT FOR DEBUG

                                ;INITIALIZE STACK

 000715  104206  000745                 MOV     #STACK,SP       ; SET UP STACK POINTER
 000717  004521                         BR      START           ; BRANCH OVER SUPPORT CODE

                                ;STACK AREA

 000720                                 .BLKW   21.                     ;STACK
 000745  123456                 STACK:  .WORD   123456                  ;MARKER FOR STACK UNDERFLOW
```

```
                                .SBTTL  RTDS   - REAL TIME DRIVE STATE ROUTINE WITH ERROR REPORTING (TEST 4)
000746  020775                  RTDS:   CALL    RTDSL           ; GET REAL TIME DRIVE STATE
000747  115002                          TST     R2              ; SEE IF ERROR OCCURRED
000750  010774                          BEQ     1$              ; IF NOT, BRANCH
000751  070756                          BMI     2$              ; IF RECEIVE ERROR, BRANCH
000752                                  CERROR  4,#ER17         ; REPORT NO CLOCKS ERROR MESSAGE
000752  104200  001604  001107                                                  MOV     #ER17,HRQ.04
000755  000761                          BR      3$              ; BRANCH
000756                          2$:     CERROR  4,#ER17A        ; REPORT RECEIVE ERRORS ERROR MESSAGE
000756  104200  001634  001107                                                  MOV     #ER17A,HRQ.04
000761  104200  044021  001105  3$:     MOV     #17.!FTLDEV+4000,HRQ.02 ; ERROR NUMBER SET
000764  104200  000764  001104          MOV     #.,HRQ.01       ; ADDRESS OF ERROR SET
000767  104200  060014  001103          MOV     #ERRMC,HRQ.RQ   ; DM REQUEST SET
000772                                  ENDERR  0
000772  114000  002230                                                  CLR     ERRPOS          ; CLEAR THE POSITION
000774  000000                  1$:     RETURN

                                .SBTTL  RTDSL  - SAME AS RTDS BUT WITHOUT ERROR REPORTING
000775  104652  000025          RTDSL:  MOV     U.MASK(R5),R2   ; MOVE MASK TO R2
000777  021007                          CALL    RDSTAT          ; GET DRIVE STATUS
001000  103201  077674                  BIC     #^C<RCVRDY!ATTN!AVAIL!RWRDY>,R1 ; CLEAR UNUSED BITS
001002  115002                          TST     R2              ; TEST STATUS FOR ERROR
001003  011006                          BEQ     2$              ; IF NO ERROR, BRANCH
001004  104201  177777                  MOV     #-1,R1          ; FLAG AS INVALID
001006  000000                  2$:     RETURN

                                .SBTTL  RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
001007                          RDSTAT:
                                ;
                                ;RETURN DRIVE STATUS
                                ;STATUS RETURNED IN DM REGISTER 1
                                ;
001007                                  PUSH    R0              ; SAVE R0
001007  100467                                                          MOV R0,-(SP)
001010  104207  024000                  MOV     #24000,R0       ; MOVE 1/2 SECOND COUNT TO R0
001012  060007                  1$:     XFC     STATUS          ; GET REAL TIME DRIVE STATE
001013  102201  000004                  BIT     #DCLOCK,R1      ; SEE IF DRIVE CLOCK IS PRESENT
001015  011027                          BEQ     3$              ; IF NOT, BRANCH
001016  102201  000400                  BIT     #RCVERR,R1      ; SEE IF RECEIVE ERROR
001020  011026                          BEQ     2$              ; IF NONE, BRANCH
001021  117407                          DEC     R0              ; DECREMENT COUNT
001022  051012                          BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
001023  104202  177777                  MOV     #177777,R2      ; MARK AS RECEIVE ERROR
001025  001027                          BR      3$              ; BRANCH OVER 'NO ERRORS'
001026  114002                  2$:     CLR     R2              ; NO ERRORS
001027                          3$:     POP     R0              ; RESTORE R0
001027  104267                                                          MOV (SP)+,R0
001030  000000                          RETURN                  ; RETURN TO CALLING MODULE
```

```
      1                                 .SBTTL  HSTTLK - TALK TO THE HOST (INTERRUPT THE HOST BEFORE 3 MIN)
      2 001031                          TLKHST:
      3                                 ;
      4                                 ;       THIS ROUTINE INTERRUPTS THE HOST WITHOUT CONVEYING ANY USEFUL
      5                                 ;       INFORMATION - ALL IT DOES IS SAY 'I'M STILL ALIVE'
      6                                 ;
      7 001031                                  REPSFT  ,,,
        001031  114000  001105                                                  CLR     HRQ.02
        001033  114000  001106                                                  CLR     HRQ.03
        001035  114000  001107                                                  CLR     HRQ.04
        001037  100467                                                                  MOV R0,-(SP)
        001040  104657  000063                                                  MOV     U.UNUM(R5),R0
        001042  105657  000050                                                  ADD     U.SUBU(R5),R0
        001044  104070  001104                                                  MOV     R0,HRQ.01
        001046  104207  060007                                                  MOV     #T4SOFT,R0
        001050  021053                                                         CALL    HOSTRQ
        001051  104267                                                                 MOV (SP)+,R0
      8 001052  000000                           RETURN
```

```
                                        .SBTTL  HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
001053                                  HOSTRQ:
                                        ;
                                        ;SEND REQUEST BUFFER TO HOST AND WAIT FOR RESPONSE.
                                        ;CLEAR ARGUMENT AREA OF OUT BUFFER IN PREPARATION
                                        ;FOR NEXT HOSTRQ CALL.
                                        ;
                                        ;INPUTS:
                                        ;       R0 - HOST REQUEST NUMBER
                                        ;       OUT BUFFER LOADED WITH DATA

001053                                          PUSH    <R0,R1,R2>
001053  100467                                                          MOV R0,-(SP)
001054  100461                                                          MOV R1,-(SP)
001055  100462                                                          MOV R2,-(SP)
001056  104070  001103                         MOV     R0,HRQ.RQ               ; STORE REQUEST NUMBER IN BUFFER
001060  104207  001103                 SNDAGN: MOV     #HRQ.RQ,R0              ; SEND BUFFER TO HOST
001062  104201  000043                         MOV     #BUFSIZ,R1              ; SIZE OF SEND BUFFER
001064  060016                               XFC     MRD                     ; SEND BUFFER
001065  115001                               TST     R1                      ; CHECK FOR ERROR
001066  051060                               BNE     SNDAGN                  ; IF ERROR, TRY AGAIN
001067  104207  001103                       MOV     #HRQ.RQ,R0              ; WAIT FOR RESPONSE FROM HOST
001071  104201  000043                       MOV     #BUFSIZ,R1              ; SIZE OR RECEIVE BUFFER
001073  060017                               XFC     MWR                     ; RECEIVE BUFFER
001074  104200  177777  001103               MOV     #-1,HRQ.RQ              ; MAKE REQUEST ILLEGAL
001077                                          POP     <R2,R1,R0>
001077  104262                                                          MOV (SP)+,R2
001100  104261                                                          MOV (SP)+,R1
001101  104267                                                          MOV (SP)+,R0
001102  000000                                  RETURN
```

```
                                ;STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS
                                ;
                                ;HRQ BUFFER - DATA TO SEND/RECEIVE TO/FROM HOST
                                ;
001103  000000  HRQ.RQ: .WORD 0          ;HOST REQUEST CODE
001104  000000  HRQ.01: .WORD 0          ;DATA ARGUMENT  1
001105  000000  HRQ.02: .WORD 0          ;DATA ARGUMENT  2
001106  000000  HRQ.03: .WORD 0          ;DATA ARGUMENT  3
001107  000000  HRQ.04: .WORD 0          ;DATA ARGUMENT  4
001110  000000  HRQ.05: .WORD 0          ;DATA ARGUMENT  5
001111  000000  HRQ.06: .WORD 0          ;DATA ARGUMENT  6
001112  000000  HRQ.07: .WORD 0          ;DATA ARGUMENT  7
001113  000000  HRQ.08: .WORD 0          ;DATA ARGUMENT  8
001114  000000  HRQ.09: .WORD 0          ;DATA ARGUMENT  9
001115  000000  HRQ.10: .WORD 0          ;DATA ARGUMENT 10
001116  000000  HRQ.11: .WORD 0          ;DATA ARGUMENT 11
001117  000000  HRQ.12: .WORD 0          ;DATA ARGUMENT 12
001120  000000  HRQ.13: .WORD 0          ;DATA ARGUMENT 13
001121  000000  HRQ.14: .WORD 0          ;DATA ARGUMENT 14
001122  000000  HRQ.15: .WORD 0          ;DATA ARGUMENT 15
001123  000000  HRQ.16: .WORD 0          ;DATA ARGUMENT 16
001124  000000  HRQ.17: .WORD 0          ;DATA ARGUMENT 17
001125  000000  HRQ.18: .WORD 0          ;DATA ARGUMENT 18
001126  000000  HRQ.19: .WORD 0          ;DATA ARGUMENT 19
001127  000000  HRQ.20: .WORD 0          ;DATA ARGUMENT 20
001130  000000  HRQ.21: .WORD 0          ;DATA ARGUMENT 21
001131  000000  HRQ.22: .WORD 0          ;DATA ARGUMENT 22
001132  000000  HRQ.23: .WORD 0          ;DATA ARGUMENT 23
001133  000000  HRQ.24: .WORD 0          ;DATA ARGUMENT 24
001134  000000  HRQ.25: .WORD 0          ;DATA ARGUMENT 25
001135  000000  HRQ.26: .WORD 0          ;DATA ARGUMENT 26
001136  000000  HRQ.27: .WORD 0          ;DATA ARGUMENT 27
001137  000000  HRQ.28: .WORD 0          ;DATA ARGUMENT 28
001140  000000  HRQ.29: .WORD 0          ;DATA ARGUMENT 29
001141  000000  HRQ.30: .WORD 0          ;DATA ARGUMENT 30
001142  000000  HRQ.31: .WORD 0          ;DATA ARGUMENT 31
001143  000000  HRQ.32: .WORD 0          ;DATA ARGUMENT 32
001144  000000  HRQ.33: .WORD 0          ;DATA ARGUMENT 33
001145  000000  HRQ.34: .WORD 0          ;DATA ARGUMENT 34

        000043  BUFSIZ  =       . - HRQ.RQ          ;SIZE OF BUFFER
```

CMPEDC - EDC CALCULATION ROUTINE

```
                                .SBTTL  CMPEDC - EDC CALCULATION ROUTINE
001146                          CMPEDC:
                                ;
                                ;       THIS WILL COMPUTE THE EDC OF A SECTOR POINTED TO BY R0, RETURNING
                                ;       THE VALUE IN R2
                                ;
001146                                  PUSH    <R4,R1,R0>      ; SAVE R1 AND R0 AND R4
001146  100464                                                                  MOV R4,-(SP)
001147  100461                                                                  MOV R1,-(SP)
001150  100467                                                                  MOV R0,-(SP)
001151  104201  000400                       MOV     #256.,R1        ; COMPUTE OVER 256 WORDS
001153  104202  000105                       MOV     #INTEDC,R2      ; MOVE INITIAL EDC VALUE TO R2
001155  104274                  EDCLOP: MOV     (R0)+,R4        ; GET A WORD
001156                                  RXOR    R4,R2           ; EXCLUSIVE OR THE WORD WITH THE CURRENT EDC VALUE
001156  100462                          MOV     R2,-(SP)        ; SAVE REGISTER R2
001157  103042                          BIC     R4,R2           ; CLEAR COORESPONDING BITS IN R2
001160  103264                          BIC     (SP)+,R4        ; CLEAR COORESPONDING BITS IN R4
001161  101042                          BIS     R4,R2           ; OR WHAT'S LEFT
001162  105022                          ADD     R2,R2           ; SHIFT R2 LEFT BY 1
001163  041165                          BCC     ZEREDC          ; IF THE HIGH BIT WAS CLEAR, BRANCH
001164  115402                          INC     R2              ; SET LO BIT TO 1 (OLD HI BIT)
001165  117401                  ZEREDC: DEC     R1              ; DECREMENT COUNT
001166  051155                          BNE     EDCLOP          ; IF COUNT UNEXPIRED, BRANCH
001167                                  POP     <R0,R1,R4>      ; RESTORE R1 AND R0 AND R4
001167  104267                                                                  MOV (SP)+,R0
001170  104261                                                                  MOV (SP)+,R1
001171  104264                                                                  MOV (SP)+,R4
001172  000000                          RETURN
```

```
                                .SBTTL  TALK   - SDI LEVEL 2 INTERCHANGE ROUTINE
001173                          TALK:
                                ;
                                ;       TALK SENDS THE COMMAND TO THE DRIVE.  IF AN ERROR OCCURRS, R2 IS
                                ;       RETURNED NONZERO
                                ;
                                ;
001173                                  PUSH    <R4,R5>         ; SAVE POINTER TO UNIT AND SUBUNIT PARAMETERS
001173  100464                                                                  MOV R4,-(SP)
001174  100465                                                                  MOV R5,-(SP)
001175  104652  000025                 MOV     U.MASK(R5),R2   ; GET UNIT SDI SELECT MASK
001177  104205  000001                 MOV     #1,R5           ; SEND TIMEOUT DEFAULT IS ONE
001201  106203  001616                 CMP     #SNDONE,R3      ; SEE IF ONLY TO BE SENT ONCE
001203  071206                         BMI     1$              ; IF SO, BRANCH
001204  104205  001750                 MOV     #MAXSND,R5      ; SEND MAXIMUM NUMBER OF TIMES (ONLINE)
001206  104137                 1$:     MOV     (R3),R0         ; POINTS TO SDI COMMAND BUFFER
001207                                  ASSUME  L2.OPC,0
001207  104631  000001                 MOV     L2.SLN(R3),R1   ; LOAD BYTE COUNT
001211  060004                         XFC     SEND            ; SEND SDI COMMAND
001212  115001                         TST     R1              ; SEE IF SDI COMMAND SENT SUCESSFULLY
001213  011237                         BEQ     2$              ; IF SO, BRANCH
001214  117405                         DEC     R5              ; DECREMENT TIMEOUT
001215  051206                         BNE     1$              ; IF UNEXPIRED, BRANCH
001216                                  POP     R5              ; RESTORE R5
001216  104265                                                                  MOV (SP)+,R5
001217                                  PUSH    R3              ; SAVE SDI PACKET POINTER
001217  100463                                                                  MOV R3,-(SP)
001220  024223                         CALL    GOINIT          ; INIT THE DRIVE
001221                                  SOFTER  70              ; REPORT
001221  104200  004261  001107                                                  MOV     #ER70,HRQ.04
001224  104202  147746                                                          MOV     #70!ERSOFT+4000.,R2
001226  104020  001105                                                          MOV     R2,HRQ.02
001230  104200  001230  001104                                                  MOV     #.,HRQ.01
001233  104200  060013  001103                                                  MOV     #ERRMES,HRQ.RQ
001236  001440                         BR      11$             ; BRANCH
001237                          2$:     POP     R5              ; RESTORE R5
001237  104265                                                                  MOV (SP)+,R5
001240  104654  000033                 MOV     U.SDIS(R5),R4   ; R4 HAS SHORT TIMEOUT
001242  106203  001654                 CMP     #LONG,R3        ; SEE IF LONG TIMEOUT TO BE USED
001244  031247                         BPL     4$              ; IF SO, BRANCH
001245  104654  000034                 MOV     U.SDIL(R5),R4   ; R4 HAS LONG TIMEOUT
001247                          4$:     PUSH    R3              ; SAVE POINTER
001247  100463                                                                  MOV R3,-(SP)
001250  020746                         CALL    RTDS            ; GET REAL TIME DRIVE STATUS
001251  115002                         TST     R2              ; OK?
001252  011255                         BEQ     5$              ; IF SO, BRANCH
001253  024201                         CALL    GORTRY          ; ELSE, RETRY
001254  001610                         BR      17$             ; AND BRANCH DUE TO ERROR
001255  110601                  5$:     ROR     R1              ; IS RCVRDY SET?
001256  041275                         BCC     6$              ; IF NOT, BRANCH
001257                                  ASSUME RCVRDY,1
001257                                  SOFTER  77
001257  104200  004523  001107                                                  MOV     #ER77,HRQ.04
001262  104202  147755                                                          MOV     #77!ERSOFT+4000.,R2
001264  104020  001105                                                          MOV     R2,HRQ.02
001266  104200  001266  001104                                                  MOV     #.,HRQ.01
001271  104200  060013  001103                                                  MOV     #ERRMES,HRQ.RQ
001274  001440                         BR      11$             ; END ERROR
```

```
001275  104207  001702           6$:   MOV     #ST,R0           ; POINT TO RECEIVE BUFFER
001277  104631  000002                 MOV     L2.RLN(R3),R1    ; NUMBER OF WORDS IN RESPONSE
001301  104652  000025                 MOV     U.MASK(R5),R2    ; STORE MASK IN R2
001303  060005                         XFC     RCV              ; RECEIVE SDI RESPONSE
001304  115001                         TST     R1               ; SEE IF SDI RESPONSE RECEIVED SUCESSFULLY
001305  011475                         BEQ     13$              ; IF SO, BRANCH
001306  106201  000001                 CMP     #1,R1            ; SEE IF TIMEOUT
001310  051335                         BNE     7$               ; IF NOT, BRANCH
001311  021031                         CALL    TLKHST           ; SAY 'I'M ALIVE' TO THE HOST
001312                                 POP     R3               ; RESTORE R3
001312  104263                                                                          MOV (SP)+,R3
001313  117404                         DEC     R4               ; DECREMENT TIMEOUT VALUE
001314  051247                         BNE     4$               ; IF TIMEOUT UNEXPIRED, BRANCH
001315                                 PUSH    R3               ; SAVE R3
001315  100463                                                                          MOV R3,-(SP)
001316  024223                         CALL    GOINIT           ; INIT THE DRIVE
001317                                 SOFTER  71               ; FLAG AS ERROR
001317  104200  004275  001107                                             MOV     #ER71,HRQ.04
001322  104202  147747                                                     MOV     #71!ERSOFT+4000.,R2
001324  104020  001105                                                     MOV     R2,HRQ.02
001326  104200  001326  001104                                             MOV     #.,HRQ.01
001331  104200  060013  001103                                             MOV     #ERRMES,HRQ.RQ
001334  001440                         BR      11$              ; BRANCH
001335  106201  000002           7$:   CMP     #2,R1            ; SEE IF FIRST WORD NOT START FRAME
001337  051356                         BNE     8$               ; IF NOT, BRANCH
001340                                 SOFTER  72               ; REPORT FIRST WORD NOT START FRAME
001340  104200  004312  001107                                             MOV     #ER72,HRQ.04
001343  104202  147750                                                     MOV     #72!ERSOFT+4000.,R2
001345  104020  001105                                                     MOV     R2,HRQ.02
001347  104200  001347  001104                                             MOV     #.,HRQ.01
001352  104200  060013  001103                                             MOV     #ERRMES,HRQ.RQ
001355  001440                         BR      11$              ; BRANCH
001356  106201  000004           8$:   CMP     #4,R1            ; SEE IF FRAMING ERROR
001360  051377                         BNE     9$               ; IF NOT, BRANCH
001361                                 SOFTER  73               ; REPORT FRAMING ERROR
001361  104200  004342  001107                                             MOV     #ER73,HRQ.04
001364  104202  147751                                                     MOV     #73!ERSOFT+4000.,R2
001366  104020  001105                                                     MOV     R2,HRQ.02
001370  104200  001370  001104                                             MOV     #.,HRQ.01
001373  104200  060013  001103                                             MOV     #ERRMES,HRQ.RQ
001376  001440                         BR      11$              ; BRANCH
001377  106201  000010           9$:   CMP     #10,R1           ; SEE IF CHECKSUM ERROR
001401  051420                         BNE     10$              ; IF NOT, BRANCH
001402                                 SOFTER  74               ; REPORT CHECKSUM ERROR
001402  104200  004366  001107                                             MOV     #ER74,HRQ.04
001405  104202  147752                                                     MOV     #74!ERSOFT+4000.,R2
001407  104020  001105                                                     MOV     R2,HRQ.02
001411  104200  001411  001104                                             MOV     #.,HRQ.01
001414  104200  060013  001103                                             MOV     #ERRMES,HRQ.RQ
001417  001440                         BR      11$              ; BRANCH
001420  106201  000020           10$:  CMP     #20,R1           ; SEE IF BUFFER TOO SMALL
001422  051447                         BNE     12$              ; IF NOT, BRANCH
001423                                 SOFTER  75               ; REPORT BUFFER TO SMALL
001423  104200  004413  001107                                             MOV     #ER75,HRQ.04
001426  104202  147753                                                     MOV     #75!ERSOFT+4000.,R2
001430  104020  001105                                                     MOV     R2,HRQ.02
001432  104200  001432  001104                                             MOV     #.,HRQ.01
```

```
                001435  104200  060013  001103                                  MOV     #ERRMES,HRQ.RQ
     73         001440                          11$:    ENDERR  6               ; FLAG END OF REPORTING BUFFER
                001440  104200  000051  001111                                  MOV     #SER22,HRQ.06
                001443  104200  000007  002230                                  MOV     #6+1,ERRPOS     ; SET THE POSITION
     74         001446  001537                  BR      14$                     ; EXIT
     75         001447                          12$:    HARDER  78              ; UNKNOWN ERROR RETURNED BY UDA
                001447  104200  004557  001107                                  MOV     #ER78,HRQ.04
                001452  104202  107756                                          MOV     #78!ERHARD+4000.,R2
                001454  104020  001105                                          MOV     R2,HRQ.02
                001456  104200  001456  001104                                  MOV     #.,HRQ.01
                001461  104200  060014  001103                                  MOV     #ERRMC,HRQ.RQ
     76         001464                          CERROR  6,R1                    ; REPORT ERROR
                001464  104010  001111                                          MOV     R1,HRQ.06
     77         001466                          ENDERR  7                       ; FLAG END OF REPORTING BUFFER
                001466  104200  000051  001112                                  MOV     #SER22,HRQ.07
                001471  104200  000010  002230                                  MOV     #7+1,ERRPOS     ; SET THE POSITION
     78         001474  001537                  BR      14$                     ; EXIT
     79         001475  114002                  13$:    CLR     R2              ; FLAG AS NO ERROR OCCURED
     80         001476  106637  000003          CMP     L2.ERS(R3),R0           ; SEE IF COMMAND ACCEPTED
     81         001500  011557                  BEQ     16$                     ; IF SO, BRANCH
     82         001501                          SOFTER  76                      ; REPORT
                001501  104200  004444  001107                                  MOV     #ER76,HRQ.04
                001504  104202  147754                                          MOV     #76!ERSOFT+4000.,R2
                001506  104020  001105                                          MOV     R2,HRQ.02
                001510  104200  001510  001104                                  MOV     #.,HRQ.01
                001513  104200  060013  001103                                  MOV     #ERRMES,HRQ.RQ
     83         001516                          CERROR  6,<L2.ERS(R3),R0,#SER22> ; REPORT FURTHER ERRORS
                001516  104630  000003  001111                                  MOV     L2.ERS(R3),HRQ.06
                001521  104070  001112                                          MOV     R0,HRQ.07
                001523  104200  000051  001113                                  MOV     #SER22,HRQ.08
     84         001526  104200  100011  002230          MOV     #<VALID+11>,ERRPOS ; FLAG AS STATUS GOOD
     85         001531  106207  000175          CMP     #UNSSUC,R0              ; SEE IF UNSUCCESSFUL RESPONSE
     86         001533  011537                  BEQ     14$                     ; IF SO, BRANCH
     87         001534  103200  100000  002230  BIC     #VALID,ERRPOS           ; GET STATUS FOR PRINTING
     88         001537  024201                  14$:    CALL    GORTRY          ; RETRY THIS COMMAND (MINIMUM)
     89         001540                          POP     R3                      ; RESTORE POINTER TO COMMAND
                001540  104263                                                                  MOV (SP)+,R3
     90         001541  104633  000004          MOV     L2.EOF(R3),R3           ; GET SDI ERROR OFFSET
     91         001543  105053                  ADD     R5,R3                   ; POINT TO ERROR WORD
     92         001544  104134                  MOV     (R3),R4                 ; GET ERROR COUNT
     93         001545  115404                  INC     R4                      ; INCREMENT COUNT
     94         001546  106204  000002          CMP     #2,R4                   ; SEE IF MAX
     95         001550  041555                  BCC     15$                     ; IF NOT, BRANCH
     96         001551  103200  100000  001105  BIC     #C2DFTL,HRQ.02          ; CHANGE ERROR TO A DEVICE FATAL
     97         001554  114004                  CLR     R4                      ; IN CASE DROPS DISABLED, CLEAR ERROR COUNT
     98         001555  100134                  15$:    MOV     R4,(R3)         ; SAVE COUNT
     99         001556  001610                  BR      17$                     ; EXIT
    100         001557                          16$:    POP     R3              ; RESTORE POINTER TO COMMAND
                001557  104263                                                                  MOV (SP)+,R3
    101         001560  104633  000004          MOV     L2.EOF(R3),R3           ; GET SDI ERROR OFFSET
    102         001562  105053                  ADD     R5,R3                   ; POINT TO ERROR WORD
    103         001563  104134                  MOV     (R3),R4                 ; GET ERROR COUNT
    104         001564  011610                  BEQ     17$                     ; IF NO RETRIES, BRANCH
    105         001565  100132                  MOV     R2,(R3)                 ; ZERO RETRIES
    106         001566                          REPSFT  SOFT,,,COMM             ; REPORT SOFT AND COMMUNICATION ERRORS
                001566  104200  000001  001105                                  MOV     #1,HRQ.02
                001571  114000  001106                                          CLR     HRQ.03
```

```
        001573  114000  001107                                         CLR     HRQ.04
        001575  100467                                                        MOV R0,-(SP)
        001576  104657  000063                                         MOV     U.UNUM(R5),R0
        001600  105657  000050                                         ADD     U.SUBU(R5),R0
        001602  104070  001104                                         MOV     R0,HRQ.01
        001604  104207  060007                                         MOV     #T4SOFT,R0
        001606  021053                                                 CALL    HOSTRQ
        001607  104267                                                        MOV (SP)+,R0
    107 001610                          17$:    POP     R4          ; RESTORE R4
        001610  104264                                                        MOV (SP)+,R4
    108 001611  000000                          RETURN
```

```
                                        .SBTTL  SDI PROTOCOL MESSAGE TABLES AND RESPONSE BUFFER
                                        ;
                                        ;       MESSAGE TABLES
                                        ;
            000000                      SDI$    =       0               ; INITIAL OFFSET
 001612                                 CR.ONL: MSG ONL,2,7,COMPLT      ; BRING DRIVE ONLINE
 001612  001700                                 .WORD   ONL                     ; ADDRESS OF COMMAND
 001613  000002                                 .WORD   2                       ; SIZE OF COMMAND IN BYTES
 001614  000007                                 .WORD   7                       ; SIZE OF REPLY IN WORDS
 001615  000176                                 .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
 001616  000035                                 .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
            001616                      SNDONE  =       .-1     ; PREVIOUS COMMANDS CAN BE SENT TO AN OFFLINE DRIVE
 001617                                 CR.GST: MSG GST,1,7,STSRES      ; GET STATUS
 001617  001670                                 .WORD   GST                     ; ADDRESS OF COMMAND
 001620  000001                                 .WORD   1                       ; SIZE OF COMMAND IN BYTES
 001621  000007                                 .WORD   7                       ; SIZE OF REPLY IN WORDS
 001622  000366                                 .WORD   STSRES                  ; SUCCESSFUL COMPLETION CODE
 001623  000036                                 .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
 001624                                 CR.CLR: MSG DRC,2,7,COMPLT      ; DRIVE CLEAR
 001624  001666                                 .WORD   DRC                     ; ADDRESS OF COMMAND
 001625  000002                                 .WORD   2                       ; SIZE OF COMMAND IN BYTES
 001626  000007                                 .WORD   7                       ; SIZE OF REPLY IN WORDS
 001627  000176                                 .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
 001630  000037                                 .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
 001631                                 CR.MOD: MSG MOD,3,7,COMPLT      ; CHANGE MODE
 001631  001664                                 .WORD   MOD                     ; ADDRESS OF COMMAND
 001632  000003                                 .WORD   3                       ; SIZE OF COMMAND IN BYTES
 001633  000007                                 .WORD   7                       ; SIZE OF REPLY IN WORDS
 001634  000176                                 .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
 001635  000040                                 .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
 001636                                 CR.SEK: MSG INS,6,7,COMPLT      ; INITIATE SEEK
 001636  001672                                 .WORD   INS                     ; ADDRESS OF COMMAND
 001637  000006                                 .WORD   6                       ; SIZE OF COMMAND IN BYTES
 001640  000007                                 .WORD   7                       ; SIZE OF REPLY IN WORDS
 001641  000176                                 .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
 001642  000041                                 .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
 001643                                 CR.ERR: MSG ERR,2,7,COMPLT      ; ERROR RECOVERY
 001643  001676                                 .WORD   ERR                     ; ADDRESS OF COMMAND
 001644  000002                                 .WORD   2                       ; SIZE OF COMMAND IN BYTES
 001645  000007                                 .WORD   7                       ; SIZE OF REPLY IN WORDS
 001646  000176                                 .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
 001647  000042                                 .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
 001650                                 CR.DIS: MSG DIS,2,7,COMPLT      ; DISCONNECT DRIVE
 001650  001662                                 .WORD   DIS                     ; ADDRESS OF COMMAND
 001651  000002                                 .WORD   2                       ; SIZE OF COMMAND IN BYTES
 001652  000007                                 .WORD   7                       ; SIZE OF REPLY IN WORDS
 001653  000176                                 .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
 001654  000043                                 .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
            001654                      LONG    =       .-1     ; ALL COMMANDS BEYOND THIS POINT ARE LONG TIMEOUT
 001655                                 CR.INR: MSG INR,1,7,COMPLT      ; INITIATE RECALIBRATE
 001655  001671                                 .WORD   INR                     ; ADDRESS OF COMMAND
 001656  000001                                 .WORD   1                       ; SIZE OF COMMAND IN BYTES
 001657  000007                                 .WORD   7                       ; SIZE OF REPLY IN WORDS
 001660  000176                                 .WORD   COMPLT                  ; SUCCESSFUL COMPLETION CODE
 001661  000044                                 .WORD   SDI$+U.SDI2             ; RETRY COUNT OFFSET
            000010                      NUML2S  =       SDI$            ; SAVE NUMBER OF LEVEL 2 SDI COMMANDS
                                        ;
```

```
                                        ;       LEVEL 2 COMMAND MESSAGE DATA STRUCTURES
                                        ;
001662    000     204           DIS:    .BYTE   0,DISCON        ; DISCONNECT DRIVE
001663  000000                          .WORD   0               ; DO NOT SPIN DOWN DRIVE
001664    000     201           MOD:    .BYTE   0,CHGMOD        ; CHANGE MODE
001665  000000                  WRITBT: .WORD   0               ; MODE
001666    000     005           DRC:    .BYTE   0,DRVCLR        ; DRIVE CLEAR
001667  000000                  DCLR:   .WORD   0
001670    000     011           GST:    .BYTE   0,GETSTA        ; GET STATUS
001671    000     216           INR:    .BYTE   0,IRECLB        ; INITIATE RECALIBRATE
001672    000     012           INS:    .BYTE   0,INSEEK        ; INITIATE SEEK
001673  000000                  LOCYL:  .WORD   0               ; INS CYLINDER/HEAD ARGUMENTS
001674  000000                          .WORD   0               ; HI CYLINDER
001675  000000                          .WORD   0               ; GROUP
001676    000     006           ERR:    .BYTE   0,ERECOV        ; ERROR RECOVERY
001677  000000                  ERRLEV: .WORD   0
001700    000     213           ONL:    .BYTE   0,DRVONL        ; ONLINE COMMAND
001701  000012                          .WORD   10.             ; TIMEOUT VALUE

001702                          ST:     .BLKW   19.             ; RESPONSE BUFFER
```

```
                                .SBTTL  BLKCHK - SEE If A BLOCK WITH ERROR IS KNOWN TO BE BAD
001725                          BLKCHK:
                                ;
                                ;       BLKCHK CHECKS THE BLOCK IN U.LBN TO ASSURE THAT IT IS NOT A BAD
                                ;       BLOCK.  IF SO, R2 IS RETURNED NONZERO
                                ;
001725                          PUSH    <R2,R1,R0>      ; SAVE ALL REG'S
001725  100462                                                          MOV R2,-(SP)
001726  100461                                                          MOV R1,-(SP)
001727  100467                                                          MOV R0,-(SP)
001730  104651  000046          MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
001732  102201  000200          BIT     #RBNBN,R1       ; SEE IF ON RBN
001734  051754                  BNE     1$              ; IF SO, UNKNOWN BAD BLOCK
001735  104641  000012          MOV     S.BADP(R4),R1   ; GET BAD BLOCK POINTER
001737  011754                  BEQ     1$              ; IF NO BAD BLOCKS, BRANCH
001740  104202  000002          MOV     #RW.LOW,R2      ; POINT TO LBN TO BE TESTED
001742  105072                  ADD     R0,R2           ; POINT TO LBN TO BE TESTED
001743  021765          3$:     CALL    CMP2            ; COMPARE BAD BLOCK TO LBN
001744  011757                  BEQ     2$              ; IF EQUAL, BRANCH
001745                          BCS     1$              ; IF LIST GREATER THAN BLOCK, IT'S OK
001745  041747                                                          BCC     .+2
001746  001754                                                          BR      1$
001747  105201  000002          ADD     #2,R1           ; POINT TO NEXT BAD BLOCK
001751  104617  177777          MOV     -1(R1),R0       ; SEE IF EOL
001753  031743                  BPL     3$              ; IF NOT, BRANCH
001754  104207  000001  1$:     MOV     #1,R0           ; SET UP FOR CARRY TO BE SET (UNKNOWN BAD BLOCK)
001756  001760                  BR      4$
001757  114007          2$:     CLR     R0              ; SET UP FOR THE CARRY TO BE CLEAR (BAD BLOCK KNOWN)
001760  110607          4$:     ROR     R0              ; SET CARRY TO REFLECT KNOWLEDGE OF BAD BLOCK
001761                          POP     <R0,R1,R2>      ; RESTORE
001761  104267                                                          MOV (SP)+,R0
001762  104261                                                          MOV (SP)+,R1
001763  104262                                                          MOV (SP)+,R2
001764  000000                  RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  CMP2    - 28 BIT COMPARE
001765                          CMP2:
                                ;
                                ;       CMP2 COMPARES A 28 BIT NUMBER POINTED TO BY R2 TO A 28 BIT NUMBER
                                ;       POINTED TO BY R1.  BOTH NUMBERS ARE LOW ORDER WORD FOLLOWED BY HIGH
                                ;       WORD (POINTER TO LOW WORD) AND THE HIGH 4 BITS <31:28> OF THE
                                ;       WORD POINTED TO BY R1 ARE STRIPPED OFF (THIS IS TO ELIMINATE THE
                                ;       END-OF-LIST FLAG ON THE B/E SETS AND BAD BLOCKS)
                                ;
001765  104617  000001                MOV     1(R1),R0        ; MOVE HIGH ORDER WORD TO R0
001767  103207  170000                BIC     #^CHBHINB,R0    ; CLEAR UNUSED BITS
001771  106627  000001                CMP     1(R2),R0        ; COMPARE HIGH ORDER WORRD TO R0
001773  051776                        BNE     1$              ; IF UNEQUAL, BRANCH
001774  104117                        MOV     (R1),R0         ; MOVE LOW ORDER WORD TO R0
001775  106127                        CMP     (R2),R0         ; COMPARE LOW ORDER WORD TO R0
001776  000000                1$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  BULDUM - BUILD THE DUMMY SDI CONTROL BLOCK FOR READS AND WRITES
001777                                  BULDUM:
                                        ;
                                        ;       BULDUM WILL BUILD THE DUMMY SDI CONTROL BLOCK
                                        ;
001777  114002                                  CLR     R2                      ; NO RBN'S
002000  104643  000000                          MOV     S.PARM(R4),R3           ; GET SUBUNIT PARAMETERS
002002  102203  020000                          BIT     #DCYLS,R3               ; SEE IF IN DIAGNOSTIC AREA
002004  052013                                  BNE     1$                      ; IF SO, BRANCH
002005  104642  000007                          MOV     S.SCHR(R4),R2           ; GET SUBUNIT CHARACTERISTICS
002007  104622  000004                          MOV     RBNTRK(R2),R2           ; GET RBN'S/TRACK
002011  103202  177600                          BIC     #HIBYTE!200,R2          ; CLEAR UNUSED BITS
002013  105642  000006              1$:         ADD     S.TRKL(R4),R2           ; GET SECTORS/TRACK
002015  105022                                  ADD     R2,R2                   ; DOUBLE SECTORS/TRACK FOR HEADER COMPARE LIMIT
002016  104020  002210                          MOV     R2,DUMSDI+D.LIMT                ; MOVE TO DUMMY SEARCH LIMIT
002020  104643  000007                          MOV     S.SCHR(R4),R3           ; R3 POINTS TO SUBUNIT CHARACTERISTICS
002022  107203  000005                          SUB     #5,R3                   ; R3 POINTS TO SUB CHAR - 5
002024  104030  002211                          MOV     R3,DUMSDI+D.SCHR                ; MOVE TO DUMMY SUB CHAR POINTER
002026  000000                                  RETURN                          ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  CALC   - CALCULATE THE CYL, GRP AND TRACK FOR THE GIVEN L/RBN
002027                                  CALC:
                                        ;
                                        ;       CALC CALCULATES THE CYLINDER, GROUP, TRACK AND SECTOR OF THE BLOCK
                                        ;
002027  115007                                  TST     R0              ; SEE IF CALCULATION AREA MUST BE SET UP
002030  052164                                  BNE     MOVOUT          ; IF NOT, BRANCH
002031  104070  002200                          MOV     R0,RBNFLG       ; SET RBN FLAG AS ZERO
002033  104643  000007                          MOV     S.SCHR(R4),R3   ; R3 POINTS TO SUBUNIT'S CHARACTERISTICS
002035  104637  000004                          MOV     RBNTRK(R3),R0   ; GET RBN'S PER TRACK
002037  103207  177600                          BIC     #HIBYTE!200,R0  ; CLEAR UNUSED BITS
002041  105637  000011                          ADD     LBNTRK(R3),R0   ; ADD LBN'S PER TRACK
002043  103207  177400                          BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
002045  104070  002217                          MOV     R0,SCR1         ; SAVE
002047  104147                                  MOV     (R4),R0         ; GET SUBUNIT PARAMETERS
002050                                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
002050  102207  020000                          BIT     #DCYLS,R0       ; SEE IF USING THE DIAGNOSTIC CYLINDERS
002052  052123                                  BNE     MOVDBN          ; IF SO, BRANCH
002053  104637  000011                          MOV     LBNTRK(R3),R0   ; MOVE LBN'S PER TRACK TO R0
002055  103207  177400                          BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
002057  104651  000046                          MOV     U.PARM(R5),R1   ; MOVE UNIT PARAMETERS TO R0
002061  102201  000200                          BIT     #RBNBN,R1       ; SEE IF BLOCK REVECTORED
002063  012101                                  BEQ     LNCYL           ; IF NOT, BRANCH
002064  104070  002200                          MOV     R0,RBNFLG       ; STORE LBN'S PER TRACK IN RBN FLAG AREA
002066  104637  000004                          MOV     RBNTRK(R3),R0   ; MOVE RBN'S PER TRACK TO R0
002070  103207  177600                          BIC     #177600,R0      ; CLEAR THE HIGHER BITS, LEAVING RBN'S/TRACK
002072  104070  002177                          MOV     R0,LRDTRK       ; MOVE RBN'S PER TRACK TO CALCULATION AREA
002074  104637  000003                          MOV     HIRBN(R3),R0    ; GET HI RBN
002076  103207  170377                          BIC     #HBLONB,R0      ; CLEAR UNUSED BITS
002100  002107                                  BR      LNCONT          ; BRANCH
002101  104070  002177                  LNCYL:  MOV     R0,LRDTRK       ; MOVE LBN'S PER TRACK TO CALCULATION AREA
002103  104637  000002                          MOV     HILBN(R3),R0    ; GET HI LBN
002105  103207  170377                          BIC     #HBLONB,R0      ; CLEAR UNUSED BITS
002107  104070  002172                  LNCONT: MOV     R0,HIBN         ; SAVE
002111  114007                                  CLR     R0              ; LOW ORDER CYLINDER IS ZERO
002112  104070  002173                          MOV     R0,STACYL       ; SAVE LO ORDER STARTING CYLINDER
002114  104637  000001                          MOV     HICYL(R3),R0    ; R0 CONTAINS HI CYLINDER BITS
002116  103207  007777                          BIC     #HBHINB,R0      ; CLEAR UNUSED BITS
002120  104070  002174                          MOV     R0,STACYL+1     ; MOVE HI STARTING CYLINDER TO UNIT PARAMETERS
002122  002164                                  BR      MOVOUT          ; BRANCH
002123  104637  000004                  MOVDBN: MOV     RBNTRK(R3),R0   ; MOVE NUMBER OF RBN'S PER TRACK TO R0
002125  103207  177600                          BIC     #177600,R0      ; CLEAR THE HIGHER BITS, LEAVING RBN'S/TRACK
002127  104631  000011                          MOV     LBNTRK(R3),R1   ; ADD LBN'S/TRK, GIVING DBN'S/TRK
002131  103201  177400                          BIC     #HIBYTE,R1      ; CLEAR UNUSED BITS
002133  105017                                  ADD     R1,R0           ; ADD LBNS/TRK TO RBNS/TRK TO GIVE DBNS/TRK
002134  104070  002177                          MOV     R0,LRDTRK       ; SAVE DBNS/TRK IN COMPUTATIONAL AREA
002136  104647  000002                          MOV     S.SDCL(R4),R0   ; GET LO ORDER STARTING DIAGNOSTIC CYLINDER
002140  104070  002173                          MOV     R0,STACYL       ; SAVE LO ORDER CYL
002142  104637  000001                          MOV     HICYL(R3),R0    ; GET HI CYLINDER BITS
002144  103207  007777                          BIC     #HBHINB,R0      ; CLEAR UNUSED BITS
002146  101647  000003                          BIS     S.SDCL+1(R4),R0 ; GET HI ORDER STARTING DIAGNOSTIC CYLINDER
002150  104070  00217?                          MOV     R0,STACYL+1     ; SAVE HI ORDER CYL
002152  104637  00000?                          MOV     HIDBN(R3),R0    ; GET HI DBN BITS
002154  110607                                  ROR     R0              ; ROTATE TO CORRECT POSITION
002155  110607                                  ROR     R0              ; ROTATE TO CORRECT POSITION
002156  110607                                  ROR     R0              ; ROTATE TO CORRECT POSITION
002157  110607                                  ROR     R0              ; ROTATE TO CORRECT POSITION
```

CALC - CALCULATE THE CYL, GRP AND TRACK FOR THE GIVEN L/RBN

```
     58 002160  103207  170377                    BIC     #HBLONB,R0      ; CLEAR UNUSED BITS
     59 002162  104070  002172                    MOV     R0,HIBN         ; SAVE
     60 002164  104207  002173          MOVOUT:   MOV     #CALCSC,R0      ; POINT TO CALCULATION AREA
     61 002166  104641  000007                    MOV     S.SCHR(R4),R1   ; POINT TO SUBUNIT CHAR TABLE
     62 002170  060020                            XFC     CVT             ; CALCULATE
     63 002171  000000                            RETURN                  ; RETURN TO CALLING PROGRAM
```

TEST 4 STORAGE AREA FOR VARIABLES

```
                                .SBTTL  TEST 4 STORAGE AREA FOR VARIABLES
002172                  HIBN:   .BLKW   1                       ; HI BN BITS <27:24>
002173                  CALCSC:                                 ; CALCULATION AREA FOR CYL, GRP, TRK
002173                  STACYL: .BLKW   2                       ; STARTING CYLINDER NUMBER
002175                  CURBN:  .BLKW   2                       ; L/R/DBN
002177                  LRDTRK: .BLKW   1                       ; SECTORS PER TRACK
002200                  RBNFLG: .BLKW   1                       ; RBN FLAG
002201                  CYL:    .BLKW   2                       ; CYLINDER
002203                  GROUP:  .BLKW   1                       ; GROUP
002204                  TRACK:  .BLKW   1                       ; TRACK
002205                  SECTOR: .BLKW   1                       ; SECTOR
002206                  INDEX:  .BLKW   1                       ; INDEX

002207                  DUMSDI: .BLKW   8.                      ; DUMMY SDI AREA

002217                  SCR1:   .BLKW   1.                      ; XFC READ AND WRITE WILL WRITE THE
002220                  SCR2:   .BLKW   1.                      ; REVECTORED LBN IN THIS SPACE,
                                                                ; OTHERWISE, USE IT FOR SCRATCH

002221                  PNUM:   .BLKW   1                       ; PATTERN NUMBER STORAGE AREA FOR ERRORS
002222                  LSTOVL: .BLKW   1                       ; LAST OVERLAY CALLED NUMBER
002223                  M.PARM: .BLKW   1                       ; MASTER PARAMETERS
002224  034245          LOSEED: .WORD   34245                   ; LO ORDER RANDOM NUMBER SEED
002225  061453          HISEED: .WORD   61453                   ; HI ORDER RANDOM NUMBER SEED
002226                  ERMODE: .BLKW   1                       ; UNEXPECTED ATTENTION FLAG
002227  007774          MEMPOL: .WORD   HIMEM                   ; MEMORY POOL FOR UNIT DATASTRUCTURES
002230                  ERRPOS: .BLKW   1                       ; RTDS AND STATUS POSITION
002231                          .ASSUME .-DUMSDI,18.
                        ; *** NEXT WORD REQUIRED BY UDA TO BE 18 PAST DUMMY SDI BLOCK START
002231  002203                  .WORD   DUMSDI-4                ; POINTER FOR LBN REVECTOR INFORMATION
                                ;       BELEIVE IT OR NOT, THIS WILL WRITE THE REVECTORED LBN
                                ;       IN LOCATIONS DUMSDI+8 AND DUMSDI+9
                                ;       EG. DUMSDI+18. POINTS TO 12 SHORT OF WHERE TO PUT THE
                                ;               REVECTOR INFORMATION

002232                  CHAINS: .BLKW   1                       ; READ/WRITE CHAIN HEADER
002233                  SECMAX: .BLKW   1                       ; MAXIMUM NUMBER OF BUFFERS THAT CAN BE READ
002234                  CHNMAX: .BLKW   1                       ; MAXIMUM NUMBER OF BUFFERS THAT CAN BE WRITTEN
002235                  MAXSEC: .BLKW   1                       ; MAXIMUM NUMBER OF SECTORS THIS PASS
```

DATA PATTERNS THAT WILL BE WRITTEN TO THE DRIVE

```
                                .SBTTL  DATA PATTERNS THAT WILL BE WRITTEN TO THE DRIVE
002236  002256  PATPTR: .WORD   PAT0                    ; POINTERS TO DATA PATTERNS
002237  002300          .WORD   PAT1
002240  002303          .WORD   PAT2
002241  002306          .WORD   PAT3
002242  002311          .WORD   PAT4
002243  002333          .WORD   PAT5
002244  002355          .WORD   PAT6
002245  002377          .WORD   PAT7
002246  002402          .WORD   PAT8
002247  002424          .WORD   PAT9
002250  002427          .WORD   PAT10
002251  002451          .WORD   PAT11
002252  002454          .WORD   PAT12
002253  002476          .WORD   PAT13
002254  002520          .WORD   PAT14
002255  002525          .WORD   PAT15
                        ;
                        ;       DATA PATTERNS (EDC OF PATTERN, LENGTH OF PATTERN IN WORDS, FOLLOWED BY PATTERN)
                        ;
002256  177777  PAT0:   .WORD   177777          ; DUMMY EDC
002257  000001          .WORD   1               ; USER DEFINEABLE PATTERN
002260  000000          .WORD   0
002261  000000          .WORD   0
002262  000000          .WORD   0
002263  000000          .WORD   0
002264  000000          .WORD   0
002265  000000          .WORD   0
002266  000000          .WORD   0
002267  000000          .WORD   0
002270  000000          .WORD   0
002271  000000          .WORD   0
002272  000000          .WORD   0
002273  000000          .WORD   0
002274  000000          .WORD   0
002275  000000          .WORD   0
002276  000000          .WORD   0
002277  000000          .WORD   0
002300  115337  PAT1:   .WORD   115337          ; EDC FOR PATTERN 1
002301  000001          .WORD   1               ; B'1000101110001011'
002302  105613          .WORD   105613
002303  010524  PAT2:   .WORD   010524          ; EDC FOR PATTERN 2
002304  000001          .WORD   1
002305  031463          .WORD   031463          ; B'0011001100110011'
002306  001747  PAT3:   .WORD   001747          ; EDC FOR PATTERN 3
002307  000001          .WORD   1
002310  030221          .WORD   030221
002311  135776  PAT4:   .WORD   135776          ; EDC FOR PATTERN 4
002312  000020          .WORD   16.             ; SHIFTING ONES
002313  000001          .WORD   000001
002314  000003          .WORD   000003
002315  000007          .WORD   000007
002316  000017          .WORD   000017
002317  000037          .WORD   000037
002320  000077          .WORD   000077
002321  000177          .WORD   000177
002322  000377          .WORD   000377
```

DATA PATTERNS THAT WILL BE WRITTEN TO THE DRIVE

```
 58 002323  000777                        .WORD   000777
 59 002324  001777                        .WORD   001777
 60 002325  003777                        .WORD   003777
 61 002326  007777                        .WORD   007777
 62 002327  017777                        .WORD   017777
 63 002330  037777                        .WORD   037777
 64 002331  077777                        .WORD   077777
 65 002332  177777                        .WORD   177777
 66 002333  052420              PAT5:     .WORD   052420          ; EDC FOR PATTERN 5
 67 002334  000020                        .WORD   16.             ; SHIFTING ZEROS
 68 002335  177776                        .WORD   177776
 69 002336  177774                        .WORD   177774
 70 002337  177770                        .WORD   177770
 71 002340  177760                        .WORD   177760
 72 002341  177740                        .WORD   177740
 73 002342  177700                        .WORD   177700
 74 002343  177600                        .WORD   177600
 75 002344  177400                        .WORD   177400
 76 002345  177000                        .WORD   177000
 77 002346  176000                        .WORD   176000
 78 002347  174000                        .WORD   174000
 79 002350  170000                        .WORD   170000
 80 002351  160000                        .WORD   160000
 81 002352  140000                        .WORD   140000
 82 002353  100000                        .WORD   100000
 83 002354  000000                        .WORD   000000
 84 002355  114734              PAT6:     .WORD   114734          ; EDC FOR PATTERN 6
 85 002356  000020                        .WORD   16.             ; ALTERNATING ZERO WORD AND ONE WORD IN
 86 002357  000000                        .WORD   000000          ; 3-2-1-1-1 SEQUENCE
 87 002360  000000                        .WORD   000000
 88 002361  000000                        .WORD   000000
 89 002362  177777                        .WORD   177777
 90 002363  177777                        .WORD   177777
 91 002364  177777                        .WORD   177777
 92 002365  000000                        .WORD   000000
 93 002366  000000                        .WORD   000000
 94 002367  177777                        .WORD   177777
 95 002370  177777                        .WORD   177777
 96 002371  000000                        .WORD   000000
 97 002372  177777                        .WORD   177777
 98 002373  000000                        .WORD   000000
 99 002374  177777                        .WORD   177777
100 002375  000000                        .WORD   000000
101 002376  177777                        .WORD   177777
102 002377  140753              PAT7:     .WORD   140753          ; EDC FOR PATTERN 7
103 002400  000001                        .WORD   1               ; B'1011011011011001'
104 002401  133331                        .WORD   133331
105 002402  021147              PAT8:     .WORD   021147          ; EDC FOR PATTERN 8
106 002403  000020                        .WORD   16.             ; B'0101010101010101' AND
107 002404  052525                        .WORD   052525          ; B'1010101010101010'
108 002405  052525                        .WORD   052525          ; IN 3-2-1-1-1 SEQURNCE
109 002406  052525                        .WORD   052525
110 002407  125252                        .WORD   125252
111 002410  125252                        .WORD   125252
112 002411  125252                        .WORD   125252
113 002412  052525                        .WORD   052525
114 002413  052525                        .WORD   052525
```

DATA PATTERNS THAT WILL BE WRITTEN TO THE DRIVE

```
    115 002414  125252                          .WORD   125252
    116 002415  125252                          .WORD   125252
    117 002416  052525                          .WORD   052525
    118 002417  125252                          .WORD   125252
    119 002420  052525                          .WORD   052525
    120 002421  125252                          .WORD   125252
    121 002422  052525                          .WORD   052525
    122 002423  125252                          .WORD   125252
    123 002424  041260                  PAT9:   .WORD   041260          ; EDC FOR PATTERN 9
    124 002425  000001                          .WORD   1               ; B'1101101101101100'
    125 002426  155554                          .WORD   155554
    126 002427  074075                  PAT10:  .WORD   074075          ; EDC FOR PATTERN 10
    127 002430  000020                          .WORD   16.             ; B'0010110100101101' AND
    128 002431  026455                          .WORD   026455          ; B'1101001011010010'
    129 002432  026455                          .WORD   026455          ; IN 3-2-1-1-1 SEQUENCE
    130 002433  026455                          .WORD   026455
    131 002434  151322                          .WORD   151322
    132 002435  151322                          .WORD   151322
    133 002436  151322                          .WORD   151322
    134 002437  026455                          .WORD   026455
    135 002440  026455                          .WORD   026455
    136 002441  151322                          .WORD   151322
    137 002442  151322                          .WORD   151322
    138 002443  026455                          .WORD   026455
    139 002444  151322                          .WORD   151322
    140 002445  026455                          .WORD   026455
    141 002446  151322                          .WORD   151322
    142 002447  026455                          .WORD   026455
    143 002450  151322                          .WORD   151322
    144 002451  153110                  PAT11:  .WORD   153110          ; EDC FOR PATTERN 11
    145 002452  000001                          .WORD   1               ; B'0110110110110110'
    146 002453  066666                          .WORD   066666
    147 002454  046211                  PAT12:  .WORD   046211          ; EDC FOR PATTERN 12
    148 002455  000020                          .WORD   16.             ; RIPLE ONE
    149 002456  000001                          .WORD   000001
    150 002457  000002                          .WORD   000002
    151 002460  000004                          .WORD   000004
    152 002461  000010                          .WORD   000010
    153 002462  000020                          .WORD   000020
    154 002463  000040                          .WORD   000040
    155 002464  000100                          .WORD   000100
    156 002465  000200                          .WORD   000200
    157 002466  000400                          .WORD   000400
    158 002467  001000                          .WORD   001000
    159 002470  002000                          .WORD   002000
    160 002471  004000                          .WORD   004000
    161 002472  010000                          .WORD   010000
    162 002473  020000                          .WORD   020000
    163 002474  040000                          .WORD   040000
    164 002475  100000                          .WORD   100000
    165 002476  121147                  PAT13:  .WORD   121147          ; EDC FOR PATTERN 13
    166 002477  000020                          .WORD   16.             ; RIPLE ZERO
    167 002500  177776                          .WORD   177776
    168 002501  177775                          .WORD   177775
    169 002502  177773                          .WORD   177773
    170 002503  177767                          .WORD   177767
    171 002504  177757                          .WORD   177757
```

DATA PATTERNS THAT WILL BE WRITTEN TO THE DRIVE

```
    172 002505  177737                        .WORD   177737
    173 002506  177677                        .WORD   177677
    174 002507  177577                        .WORD   177577
    175 002510  177377                        .WORD   177377
    176 002511  176777                        .WORD   176777
    177 002512  175777                        .WORD   175777
    178 002513  173777                        .WORD   173777
    179 002514  167777                        .WORD   167777
    180 002515  157777                        .WORD   157777
    181 002516  137777                        .WORD   137777
    182 002517  077777                        .WORD   077777
    183 002520  125677              PAT14:    .WORD   125677      ; EDC FOR PATTERN 14
    184 002521  000003                        .WORD   3           ; B'1101101101101101'
    185 002522  155555                        .WORD   155555      ; B'1011011011011011' PATTERNS 14 AND 15
    186 002523  133333                        .WORD   133333      ; B'1101101101101101' MAKE TEST 4 FORMATTER
    187 002524  155555                        .WORD   155555      ; REPEATED            COMPATIBLE
    188 002525  030206              PAT15:    .WORD   030206      ; EDC FOR PATTERN 15
    189 002526  000003                        .WORD   3           ; B'1101101101101101'
    190 002527  155555                        .WORD   155555      ; B'1011011011011011'
    191 002530  133333                        .WORD   133333      ; B'0110110110110110'
    192 002531  066666                        .WORD   066666      ; REPEATED
```

```
                                        .SBTTL  ROOT    - MAIN DRIVING MODULE OF TEST 4

002532  104205  007702        ROOT:     MOV     #FIRSTU,R5              ; R5 POINTS TO FIRST UNIT TO BE TESTED
002534  114007                          CLR     R0                      ; COUNT OF DROPPED UNITS IS SET TO ZERO
002535  104155                ROOTLP:   MOV     (R5),R5                 ; R5 POINTS TO NEXT UNIT IN CHAIN
002536                                  ASSUME  U.NEXT,0
002536  104651  000046                  MOV     U.PARM(R5),R1           ; GET UNIT PARAMETERS
002540  072560                          BMI     NOUNIT                  ; IF UNIT DROPPED, BRANCH
002541                                  ASSUME  DROP,100000             ; ASSUME DROP IS SIGN BIT
002541  115000  002223                  TST     M.PARM                  ; SEE IF IN PROCESS OF INITIAL WRITE
002543  032547                          BPL     1$                      ; IF NOT, BRANCH
002544                                  ASSUME  IWIPRG,100000           ; ASSUME IWIPRG IS SIGN BIT
002544  102201  040000                  BIT     #INITW,R1               ; SEE IF THIS UNIT IS BEING INITALLY WRITTEN
002546  012560                          BEQ     NOUNIT                  ; IF NOT, BRANCH
002547  022577                1$:       CALL    SEQNCR                  ; CALL SEQUENCER
002550  104657  000046                  MOV     U.PARM(R5),R0           ; SEE IF JUST DROPPED
002552  032556                          BPL     2$                      ; IF NOT, BRANCH
002553                                  ASSUME  DROP,100000
002553  104203  001650                  MOV     #CR.DIS,R3              ; POINT TO DISCONNECT
002555  021173                          CALL    TALK                    ; SEND
002556  114007                2$:       CLR     R0                      ; COUNT OF DROPPED UNITS IS SET TO ZERO
002557  002535                          BR      ROOTLP                  ; BRANCH TO BOTTOM OF ROOT
002560  115407                NOUNIT:   INC     R0                      ; INCREMENT COUNT OF DROPPED UNITS
002561  106207  000004                  CMP     #4,R0                   ; SEE IF FOUR (ALL) UNITS HAVE BEEN DROPPED
002563  052535                          BNE     ROOTLP                  ; IF NOT, BRANCH
002564  115000  002223                  TST     M.PARM                  ; SEE IF INITIAL WRITE WAS IN PROGRESS
002566  032573                          BPL     STOP                    ; IF NOT, END TEST4
002567                                  ASSUME  IWIPRG,100000           ; ASSUME IWIPRG IS SIGN BIT
002567  103200  100000  002223          BIC     #IWIPRG,M.PARM          ; CLEAR INITIAL WRITE BIT
002572  002532                          BR      ROOT                    ; NOW START TESTING
002573  104207  060016        STOP:     MOV     #DONE,R0                ; MOVE DONE TO HOST COMMUNICATION AREA
002575  021053                          CALL    HOSTRQ                  ; SENT DONE MESSAGE TO HOST
002576  060021                          XFC     EXIT                    ; EXIT DIAGNOSTIC MACHINE MODE
```

```
      1                                 .SBTTL  SEQNCR - OVERLAY DRIVER FOR TEST 4
      2 002577                          SEQNCR:
      3                                 ;
      4                                 ;       SEQUENCER CALLS THE VARIOUS MODULES, USING THE ADDRESS RETURNED
      5                                 ;       IN R0 BY THE PREVIOUS MODULE.  R0 HOLDS THE NEXT ADDRESS TO VECTOR
      6                                 ;       TO. IF R1 IS ZERO, AN IMMIDATE CALL IS MADE TO THE NEXT ROUTINE.
      7                                 ;       IF R2 IS NONZERO, AN ERROR WAS ENCOUNTERED, AND THE ERROR COUNT IS
      8                                 ;       INCREMENTED.  THE ERROR IS THEN REPORTED TO THE HOST.
      9                                 ;
     10 002577  104657  000013                  MOV     U.NFUN(R5),R0   ; LOAD R0 WITH NEXT FUNCTION ADDRESS
     11 002601                          SEQLP:
```

```
                                .SBTTL  DRPTST - SEE THAT AT LEAST ONE SUBUNIT IS ACTIVE ON A UNIT
                                ;DRPTST
                                ;
                                ;       DROP TEST MAKES SURE AT LEAST ONE ACTIVE SUBUNIT IS ACTIVE ON THE UNIT
                                ;       IF SO, R4 POINTS TO SUBUNIT PARAMETER BLOCK, U.SUBU IS UPDATED TO
                                ;       THE SUBUNIT OFFSET, AND R3 IS RETURNED POSITIVE.  IF NOT, R3 IS
                                ;       RETURNED NEGATIVE AND THE ENTIRE UNIT IS DROPPED
                                ;
 002601                         DROP:   PUSH    R0              ; SAVE NEXT MODULE ADDRESS
 002601  100467                                                                  MOV R0,-(SP)
 002602  104657  000046                 MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
 002604  102207  001000                 BIT     #FTIME,R0       ; SEE IF FIRST TIME THROUGH
 002606  012673                        BEQ     1$              ; IF NOT, BRANCH
 002607  024207                        CALL    GOSEEK          ; IF FIRST TIME, FORCE A SEEK
 002610  103207  015610                BIC     #RBNBN!REVEC!FTIME!DIREC!WCHECK!NEWSUB,R0
 002612  100657  000046                MOV     R0,U.PARM(R5)   ; SAVE
 002614  104057                        MOV     R5,R0           ; R0 WILL POINT AT SUBUNIT POINTERS
 002615  115407                        INC     R0              ; R0 POINTS AT SUBUNIT POINTERS
 002616                                ASSUME  U.SUBP,1
 002616  114002                        CLR     R2              ; UP TO 4 SUBUNITS ON THIS UNIT
 002617  100652  000024                MOV     R2,U.CSEC(R5)   ; ZERO ALL TRACK COUNTS SO THE TEST AT THE
 002621  100652  000021                MOV     R2,U.NSEC(R5)   ; BEGINNING OF SETUP IS SATISIFIED SO CONTROL
 002623  100652  000023                MOV     R2,U.TSEC(R5)   ; DROPS THROUGH TO SETUP THE NEXT OPERATION
 002625  104273                11$:    MOV     (R0)+,R3        ; GET SUBUNIT POINTER
 002626  072637                        BMI     12$             ; IF NO SUBUNIT, BRANCH
 002627  104133                        MOV     (R3),R3         ; GET SUBUNIT PARAMETERS
 002630                                ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
 002630  072637                        BMI     12$             ; IF DROPPED, BRANCH
 002631                                ASSUME  DROP,100000     ; ASSUME DROP IS SIGN BIT
 002631  115000  002223                TST     M.PARM          ; SEE IF INTIIAL WRITE IN PROGRESS
 002633  032644                        BPL     13$             ; IF NOT, BRANCH
 002634                                ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
 002634  102203  040000                BIT     #INITW,R3       ; SEE IF THIS SUBUNIT IS TO BE INITIALLY WRITTEN
 002636  052644                        BNE     13$             ; IF SO, BRANCH
 002637  115402                12$:    INC     R2              ; INCREMENT COUNT
 002640  106202  000003                CMP     #3,R2           ; SEE IF COUNT EXHAUSTED
 002642  032625                        BPL     11$             ; IF NOT, BRANCH
 002643  003037                        BR      4$              ; IF SO, DROP ENTIRE UNIT
 002644  100652  000050        13$:    MOV     R2,U.SUBU(R5)   ; SAVE SUBUNIT OFFSET
 002646  104024                        MOV     R2,R4           ; MOVE TO SUBUNIT POINTER REGISTER
 002647  105054                        ADD     R5,R4           ; ADD UNIT POINTER
 002650  105204  000001                ADD     #U.SUBP,R4      ; ADD SUBUNIT POINTER OFFSET
 002652  104144                        MOV     (R4),R4         ; R4 NOW POINTS TO SUBUNIT
 002653  023700                        CALL    RECOVR          ; SET UP UNIT FOR RUNNING
 002654  104207  104000                MOV     #ERMASK!SEKREQ,R0 ; DISABLE RECOVERY, FORCE A SEEK
 002656  100657  000047                MOV     R0,U.RCOV(R5)   ; SAVE IN RECOVERY WORD
 002660  104207  000010                MOV     #NUML2S,R0      ; GET NUMBER OF LEVEL 2 COMMANDS
 002662  104051                        MOV     R5,R1           ; R1 POINTS TO UNIT PARAMETERS
 002663  105201  000035                ADD     #U.SDI2,R1      ; R1 POINTS TO SDI LEVEL 2 ERROR COUNT AREA
 002665  114002                        CLR     R2              ; SET UP TO CLEAR AREA
 002666  100212                14$:    MOV     R2,(R1)+        ; CLEAR
 002667  117407                        DEC     R0              ; DECREMENT COUNT
 002670  052666                        BNE     14$             ; IF INCOMPLETE, BRANCH
 002671  114003                        CLR     R3              ; PROCESS THIS UNIT
 002672  003045                        BR      5$              ; EXIT
 002673  104054                1$:     MOV     R5,R4           ; GET POINTER TO UNIT DATASTRUCTURE
 002674  115404                        INC     R4              ; POINT AT SUBUNIT POINTERS
```

DRPTST - SEE THAT AT LEAST ONE SUBUNIT IS ACTIVE ON A UNIT

```
                                                ASSUME  U.SUBP,1
002675  105654  000050                          ADD     U.SUBU(R5),R4   ; R4 POINTS AT SPECIFIC SUBUNIT POINTER
002677  104144                                  MOV     (R4),R4         ; R4 POINTS AT SUBUNIT PARAMETERS
002700  072756                                  BMI     25$             ; NO SUBUNIT (SOMETIMES HAPPENS IN INITIALIZATION)
002701  104141                                  MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
002702                                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
002702  102207  004000                          BIT     #NEWSUB,R0      ; SEE IF SUPPOSED TO GO TO NEXT SUBUNIT
002704  012750                                  BEQ     6$              ; IF NOT, BRANCH
002705  103207  004000                          BIC     #NEWSUB,R0      ; CLEAR NEWSUB BIT
002707  100657  000046                          MOV     R0,U.PARM(R5)   ; SAVE
002711  114002                                  CLR     R2              ; FOR CEARING FOLLOWING WORDS
002712  100652  000024                          MOV     R2,U.CSEC(R5)   ; ZERO ALL TRACK COUNTS SO THE TEST AT THE
002714  100652  000021                          MOV     R2,U.NSEC(R5)   ; BEGINNING OF SETUP IS SATISIFIED SO CONTROL
002716  100652  000023                          MOV     R2,U.TSEC(R5)   ; DROPS THROUGH TO SETUP THE NEXT OPERATION
002720  115000  002223                          TST     M.PARM          ; SEE IF INITIAL WRITE IS IN PROGRESS
002722  032753                                  BPL     2$              ; IF NOT, BRANCH
002723                                          ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
002723  103201  040000                          BIC     #INITW,R1       ; CLEAR INITIAL WRITE BIT
002725  100141                                  MOV     R1,(R4)         ; SAVE
002726                                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
002726                                          MSSG    1               ; REPORT INITIAL WRITE COMPLETE
002726  104200  005527  001105                                                  MOV     #MS1,HRQ.02
002731  100467                                                                          MOV R0,-(SP)
002732  104650  000063  001104                                                  MOV     U.UNUM(R5),HRQ.01
002735  105650  000050  001104                                                  ADD     U.SUBU(R5),HRQ.01
002740  104207  060015                                                          MOV     #MESSAG,R0
002742  021053                                                                  CALL    HOSTRQ
002743  104267                                                                          MOV (SP)+,R0
002744  104652  000050                          MOV     U.SUBU(R5),R2   ; GET ACTIVE SUBUNIT POINTER
002746  024114                                  CALL    CHKUP           ; SEE IF ANY MORE UNITS TO INITIAL WRITE
002747  003045                                  BR      5$              ; BRANCH
002750  114003                          6$:     CLR     R3              ; IF ACTIVE, FLAG TO PROCESS
002751  115001                                  TST     R1              ; SEE IF UNIT ACTIVE
002752  033052                                  BPL     7$              ; IF ACTIVE, BRANCH
002753                                          ASSUME  DROP,100000     ; ASSUME DROP IS SIGN BIT
002753  102201  040100                  2$:     BIT     #SEQSEK!INITW,R1        ; SEE IF SEQUENTIAL SEEKS
002755  053015                                  BNE     10$             ; IF SO, BRANCH
                                        ;RNDSUB
                                        ;
                                        ;       GET NEXT SUBUNIT TO CHECK RANDOMLY
                                        ;
002756  104207  000004                  25$:    MOV     #4,R0           ; SUBUNIT COUNT
002760  104203  000001                          MOV     #U.SUBP,R3      ; R3 WILL POINT TO SUBUNIT POINTERS
002762  105053                                  ADD     R5,R3           ; R3 POINTS TO SUBUNIT POINTERS
002763  104652  000050                          MOV     U.SUBU(R5),R2   ; R2 IS SUBUNIT NUMBER
002765  115402                          21$:    INC     R2              ; GO TO NEXT UNIT
002766  106202  000003                          CMP     #3,R2           ; SEE IF GREATER THAN 3
002770  032772                                  BPL     22$             ; IF NOT, BRANCH
002771  114002                                  CLR     R2              ; NOW ON SUBUNIT 0
002772  104034                          22$:    MOV     R3,R4           ; R4 POINTS TO SUBUNIT POINTERS
002773  105024                                  ADD     R2,R4           ; R4 NOW POINTS TO A SPECIFIC SUBUNIT POINTER
002774  104144                                  MOV     (R4),R4         ; R4 NOW POINTS TO SUBUNIT PARAMETERS
002775  073000                                  BMI     23$             ; IF NO SUBUNIT, BRANCH
002776  104141                                  MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
002777                                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
002777  033011                                  BPL     24$             ; IF ACTIVE, BRANCH
003000  117407                          23$:    DEC     R0              ; DECREMENT COUNT
```

```
107 003001  052765                        BNE     21$             ; IF INCOMPLETE, BRANCH
108 003002  104653  000046                MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
109 003004  101203  100000                BIS     #DROP,R3        ; SET DROP BIT
110 003006  100653  000046                MOV     R3,U.PARM(R5)   ; SAVE
111 003010  003045                        BR      5$              ; EXIT
112 003011  100652  000050          24$:  MOV     R2,U.SUBU(R5)   ; SAVE SUBUNIT NUMBER
113 003013  114003                        CLR     R3              ; TEST THIS SUBUNIT IMMIDATELY
114 003014  003045                        BR      5$              ; EXIT
115 003015  104652  000050          10$:  MOV     U.SUBU(R5),R2   ; R2 IS SUBUNIT NUMBER (0-3)
116 003017  102207  010000                BIT     #DIREC,R0       ; SEE IF GOING UP OR DOWN
117 003021  053031                        BNE     3$              ; IF DOWN, BRANCH
118 003022  024114                        CALL    CHKUP           ; FIND NEXT UPPER SUBUNIT
119 003023  115007                        TST     R0              ; FIND ONE?
120 003024  013045                        BEQ     5$              ; IF SO, BRANCH
121 003025  024064                        CALL    CHKDN           ; SEARCH DOWN FOR ONE
122 003026  115007                        TST     R0              ; FIND ONE?
123 003027  013045                        BEQ     5$              ; IF SO, BRANCH
124 003030  003037                        BR      4$              ; DROP UNIT
125 003031  024064                  3$:   CALL    CHKDN           ; LOOK DOWN FOR SUBUNIT
126 003032  115007                        TST     R0              ; FIND ONE?
127 003033  013045                        BEQ     5$              ; IF SO, BRANCH
128 003034  024114                        CALL    CHKUP           ; LOOK UP FOR ONE
129 003035  115007                        TST     R0              ; FIND ONE?
130 003036  013045                        BEQ     5$              ; IF SO, BRANCH
131 003037  104653  000046          4$:   MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
132 003041  101203  100000                BIS     #DROP,R3        ; DROP UNIT
133 003043  100653  000046                MOV     R3,U.PARM(R5)   ; SAVE
134 003045  115003                  5$:   TST     R3              ; SEE IF THIS SUBUNIT TO BE TESTED
135 003046  053052                        BNE     7$              ; IF NOT, BRANCH
136 003047  104200  177777  002217        MOV     #-1,SCR1        ; FLAG AS NEW SUBUNIT
137 003052                          7$:   POP     R0              ; RESTORE R0
    003052  104267                                                                MOV (SP)+,R0
138 003053  115003                        TST     R3              ; SEE IF ACTIVE SUBUNIT EXISTS
139 003054  053675                        BNE     NOSUB           ; IF SO, BRANCH
140 003055  104070  002222                MOV     R0,LSTOVL       ; SAVE THIS OVERLAY NUMBER
144 003057  106207  000100                CMP     #100,R0         ; SEE IF THE ROUTINE IS ON AN OVERLAY
145 003061  073214                        BMI     GO4IT           ; IF NOT, BRANCH
146 003062  104071                        MOV     R0,R1           ; SAVE THE OVERLAY NUMBER
147 003063  105077                        ADD     R0,R0           ; MULTIPLY THE OVERLAY NUMBER BY 4
148 003064  105077                        ADD     R0,R0
149 003065  104672  004257                MOV     OTABLE(R0),R2   ; GET THE OVERLAY AREA NUMBER
150 003067  105022                        ADD     R2,R2           ; MULTIPLY THE AREA NUMBER BY 2
151 003070  106621  004245                CMP     PTABLE(R2),R1   ; SEE IF THE OVERLAY REQUESTED IS IN MEMORY
152 003072  013175                        BEQ     LOADED          ; IF SO, BRANCH
```

```
                                .SBTTL  OVRLAY - OVERLAY PROCESS FOR BRINGING IN OVERLAYS IF NEEDED
                                ;OVRLAY
                                ;
                                ;       BRING IN THE OVERLAY
                                ;
 003073                         41$:    PUSH    <R5,R0,R1,R2,R3>        ; SAVE REGISTERS
 003073  100465                                                                 MOV R5,-(SP)
 003074  100467                                                                 MOV R0,-(SP)
 003075  100461                                                                 MOV R1,-(SP)
 003076  100462                                                                 MOV R2,-(SP)
 003077  100463                                                                 MOV R3,-(SP)
 003100  104205  000003                 MOV     #3,R5           ; TRY TO READ 3 TIMES
 003102  104671  004262                 MOV     OTABLE+3(R0),R1 ; GET THE HI ORDER BITS OF UNIBUS ADDRESS
 003104  103201  177774                 BIC     #177774,R1      ; CLEAR UNUSED BITS
 003106  104623  004246                 MOV     PTABLE+1(R2),R3 ; POINT TO WHERE TO LOAD OVERLAY
 003110  104672  004262                 MOV     OTABLE+3(R0),R2 ; GET NUMBER OF WORDS TO TRANSFER
 003112  110602                         ROR     R2              ; ROTATE BITS TO CORRECT POSITION
 003113  110602                         ROR     R2
 003114  103202  140000                 BIC     #140000,R2      ; CLEAR UNUSED BITS
 003116  104677  004261                 MOV     OTABLE+2(R0),R0 ; GET LO ORDER UNIBUS ADDRESS
 003120  060013                         XFC     UREAD           ; READ THE OVERLAY INTO UDA MEMORY
 003121  101071                         BIS     R0,R1           ; ADD THE ERROR REGISTERS TOGETHER
 003122  013166                         BEQ     42$             ; IF NO ERRORS, BRANCH
 003123                                 POP     <R3,R2,R1,R0>   ; RESTORE THE REGISTERS
 003123  104263                                                                 MOV (SP)+,R3
 003124  104262                                                                 MOV (SP)+,R2
 003125  104261                                                                 MOV (SP)+,R1
 003126  104267                                                                 MOV (SP)+,R0
 003127                                 PUSH    <R0,R1,R2,R3>   ; SAVE THE REGISTERS
 003127  100467                                                                 MOV R0,-(SP)
 003130  100461                                                                 MOV R1,-(SP)
 003131  100462                                                                 MOV R2,-(SP)
 003132  100463                                                                 MOV R3,-(SP)
 003133  117405                         DEC     R5              ; DECREMENT COUNT
 003134  053102                         BNE     41$             ; IF INCOMPLETE, BRANCH
 003135                                 DEVFTL  33,<R1,#SER39>  ; UNABLE TO READ FROM HOST MEMORY
 003135  104200  002452  001107                                         MOV     #ER33,HRQ.04
 003140  104010  001110                                                 MOV     R1,HRQ.05
 003142  104200  002477  001111                                         MOV     #SER39,HRQ.06
 003145  104202  047701                                                 MOV     #33!FTLDEV+4000.,R2
 003147  104020  001105                                                 MOV     R2,HRQ.02
 003151  104200  003151  001104                                         MOV     #.,HRQ.01
 003154  104200  060014  001103                                         MOV     #ERRMC,HRQ.RQ
 003157  104200  177777  001106         MOV     #177777,HRQ.03  ; FLAG AS NOT ASSOCIATED WITH ANY UNIT
 003162  104307  001103                 MOV     HRQ.RQ,R0       ; SET UP TO REPORT ERROR
 003164  021053                         CALL    HOSTRQ          ; REPORT
 003165  002573                         BR      STOP            ; REPORT THAT PROGRAM IS FINISHED
 003166                         42$:    POP     <R3,R2,R1,R0,R5>        ; RESTORE THE REGISTERS
 003166  104263                                                                 MOV (SP)+,R3
 003167  104262                                                                 MOV (SP)+,R2
 003170  104261                                                                 MOV (SP)+,R1
 003171  104267                                                                 MOV (SP)+,R0
 003172  104265                                                                 MOV (SP)+,R5
 003173  100621  004245                 MOV     R1,PTABLE(R2)   ; MARK THE OVERLAY AS IN MEMORY
 003175  104677  004260         LOADED: MOV     OTABLE+1(R0),R0 ; GET THE NUMBER OF OVERLAY AREAS DESTROYED
 003177  104201  177777                 MOV     #-1,R1          ; R1 CONTAINS THE 'EMPTY AREA' FLAG
 003201                                 PUSH    R2              ; SAVE R2
```

```
                003201  100462                                                                  MOV R2,-(SP)
     33         003202  117407          GONE:   DEC     R0              ; DECREMENT THE DESTROYED COUNT
     34         003203  073211                  BMI     PLOAD           ; IF ALL NECESSARY AREAS MARKED DESTROYED, BRANCH
     35         003204  105202  000002          ADD     #2,R2           ; POINT TO NEXT OVERLAY AREA
     36         003206  100621  004245          MOV     R1,PTABLE(R2)   ; MARK AREA AS DESTROYED
     37         003210  003202                  BR      GONE            ; GO TO TOP OF LOOP
     38         003211                  PLOAD:  POP     R2              ; RESTORE R2
                003211  104262                                                                  MOV (SP)+,R2
     39         003212  104627  004246          MOV     PTABLE+1(R2),R0 ; GET OVERLAY ADDRESS
     43         003214  104652  000047  GO4IT:  MOV     U.RCOV(R5),R2   ; GET ERROR RECOVERY WORD
     44         003216  073227                  BMI     2$              ; IF NO RECOVERY, BRANCH
     45         003217                          ASSUME  ERMASK,100000   ; ASSUME ERMASK SIGN BIT
     46         003217  020746                  CALL    RTDS            ; BEFORE OPERATION, SEE IF ONLINE
     47         003220  115002                  TST     R2              ; SEE IF ERROR OCCURRED
     48         003221  053225                  BNE     1$              ; IF SO, BRANCH
     49         003222  102201  000102          BIT     #AVAIL!ATTN,R1  ; SEE IF AVAILABLE OR ATTENTION ASSERTED
     50         003224  013227                  BEQ     2$              ; IF NOT, BRANCH
     51         003225  024201          1$:     CALL    GORTRY          ; RETRY THIS MODULE
     52         003226  003231                  BR      JMPRET          ; RECOVER
     53         003227                  2$:
```

JMPRET - THIS IS THE LOCATION THAT VECTOR: TO/FROM ALL OVERLAY

```
      1                                 .SBTTL  JMPRET - THIS IS THE LOCATION THAT VECTORS TO/FROM ALL OVERLAYS
      2 003227                                  PUSH    R0              ; PUSH THE BRANCH ADDRESS ONTO THE STACK
        003227  100467                                                              MOV R0,-(SP)
      3 003230  000000                          RETURN                  ; VECTOR TO MODULE
      4 003231                          JMPRET:
```

```
                                .SBTTL  CORECT - CORRECT THE ERRORS
                                ;CORECT
                                ;
                                ;       CORRECT THE ERRORS AND RECOVER
                                ;

003231                          PUSH    <R1,R0>         ; SAVE NEXT OVERLAY AND CALL TYPE
003231  100461                                                         MOV R1,-(SP)
003232  100467                                                         MOV R0,-(SP)
003233  114000  002226          CLR     ERMODE          ; CLEAR ERROR MODES
003235  115002                  TST     R2              ; SEE IF ERROR OR MBYTE REPORT
003236  053331                  BNE     5$              ; IF SO, BRANCH
003237  104652  000047          MOV     U.RCOV(R5),R2   ; GET ERROR RECOVERY WORD
003241  073671                  BMI     90$             ; IF RECOVERY DISABLED, BRANCH
003242                          ASSUME  ERMASK,100000   ; ASSUME RECOVERY MASK IS SIGN BIT
003242  020746                  CALL    RTDS            ; GET REAL TIME DRIVE STATE
003243  115002                  TST     R2              ; SEE IF ERROR OCCURRED
003244  053361                  BNE     70$             ; IF SO, REPORT
003245  102201  000100          BIT     #AVAIL,R1       ; SEE IF AVAILABLE ASSERTED
003247  013277                  BEQ     2$              ; IF NOT, BRANCH
003250  102201  000002          BIT     #ATTN,R1        ; SEE IF SPINABLE
003252  053273                  BNE     1$              ; IF SO, BRANCH
003253                          DEVFTL  13              ; REPORT UNSPINABLE DRIVE
003253  104200  001406  001107                                         MOV     #ER13,HRQ.04
003256  104202  047655                                                 MOV     #13!FTLDEV+4000.,R2
003260  104020  001105                                                 MOV     R2,HRQ.02
003262  104200  003262  001104                                         MOV     #.,HRQ.01
003265  104200  060014  001103                                         MOV     #ERRMC,HRQ.RQ
003270                          ENDERR  0
003270  114000  002230                                                 CLR     ERRPOS          ; CLEAR THE POSITION
003272  003361                  BR      70$             ; REPORT
003273  101200  000100  002226  1$:     BIS     #AVAIL,ERMODE   ; AVAIL ERROR
003276  003302                  BR      3$              ; BRANCH
003277  102201  000002          2$:     BIT     #ATTN,R1        ; SEE IF ATTENTION IS ASSERTED
003301  013671                  BEQ     90$             ; IF NOT, BRANCH
003302  101200  000002  002226  3$:     BIS     #ATTN,ERMODE    ; FLAG AS ATTENTION ERROR
003305                          SOFTER  2               ; ATTENTION ASSERTED UNEXPECTEDLY
003305  104200  000117  001107                                         MOV     #ER2,HRQ.04
003310  104202  147642                                                 MOV     #2!ERSOFT+4000.,R2
003312  104020  001105                                                 MOV     R2,HRQ.02
003314  104200  003314  001104                                         MOV     #.,HRQ.01
003317  104200  060013  001103                                         MOV     #ERRMES,HRQ.RQ
003322                          ENDERR
003322  104200  000051  001110                                         MOV     #SER22,HRQ.05
003325  104200  000006  002230                                         MOV     #5+1,ERRPOS     ; SET THE POSITION
003330  003364                  BR      6$              ; BRANCH
```

```
      1 003331  106200  060011  001103  5$:     CMP     #T4MXFR,HRQ.RQ  ; SEE IF MEGABIT TRANSFER REPORT
      2 003334  053352                          BNE     40$             ; IF NOT, BRANCH
      3 003335  104650  000063  001104          MOV     U.UNUM(R5),HRQ.01       ; MOVE STARTING UNIT NUMBER TO COM AREA
      4 003340  105650  000050  001104          ADD     U.SUBU(R5),HRQ.01       ; ADD OFFSET
      5 003343  104307  001103                  MOV     HRQ.RQ,R0       ; SETUP FOR COM
      6 003345  021053                          CALL    HOSTRQ          ; REPORT
      7 003346  115000  001104                  TST     HRQ.01          ; SEE IF DROPPED
      8 003350  073636                          BMI     80$             ; IF SO, BRANCH
      9 003351  003671                          BR      90$             ; IF NOT, BRANCH
```

```
003352  104302  001105          40$:    MOV     HRQ.02,R2         ; GET ERROR TYPE
003354  103202  037777                  BIC     #37777,R2         ; CLEAR ERROR NUMBER
003356  106202  040000                  CMP     #FTLDEV,R2        ; SEE IF DEVICE FATAL
003360  053364                          BNE     6$                ; IF NOT, BRANCH
003361                                  ASSUME  FTLDEV,040000     ; ASSUME IN HI 2 BITS
003361  101200  100000  002226  70$:    BIS     #DROP,ERMODE      ; ERROR CAUSED THE ENTIRE DEVICE TO BE DROPPED
003364  104307  002230          6$:     MOV     ERRPOS,R0         ; GET ERROR POSITION
003366  013523                          BEQ     20$               ; IF NO REPORTING, BRANCH
003367  103207  100000                  BIC     #VALID,R0         ; CLEAR 'ALLREADY VALID' BIT
003371  105207  001104                  ADD     #HRQ.RQ+1,R0      ; POINT TO OUTPUT BUFFER
003373                                  PUSH    <R4,R0>           ; SAVE SUBUNIT POINTER AND ERROR POINTER
003373  100464                                                                          MOV R4,-(SP)
003374  100467                                                                          MOV R0,-(SP)
003375  104657  000011                  MOV     U.MSTO(R5),R0     ; GET MASTER SEEK TIMEOUT
003377  115407                          INC     R0                ; INCREMENT R0 FOR BNE RATHER THAN BCC
003400  114003                          CLR     R3                ; FOR LOW ORDER TIMEOUT
003401  020775                  31$:    CALL    RTDSL             ; GET REAL TIME DRIVE STATE
003402  115002                          TST     R2                ; SEE IF ERROR
003403  053417                          BNE     33$               ; IF ERROR, BRANCH
003404  102201  000003                  BIT     #RCVRDY!ATIN,R1   ; SEE IF RECEIVER READY OR ATTENTION ASSERTED
003406  053417                          BNE     33$               ; IF SO, BRANCH
003407  104201  000310                  MOV     #200.,R1          ; INNER LOOP TIMEOUT
003411  117401                  32$:    DEC     R1                ; DECREMENT COUNT
003412  053411                          BNE     32$               ; IF INCOMPLETE, BRANCH
003413  117403                          DEC     R3                ; DECREMENT TIMEOUT
003414  053401                          BNE     31$               ; LOOP IF INCOMPLETE
003415  117407                          DEC     R0                ; DECREMENT TIMEOUT
003416  053401                          BNE     31$               ; IF INCOMPLETE, BRANCH
003417  104657  000047          33$:    MOV     U.RCOV(R5),R0     ; GET RECOVERY TYPE
003421  102207  040000                  BIT     #DRINIT,R0        ; SEE IF DRIVE WAS INITIALIZED
003423  013432                          BEQ     34$               ; IF NOT, BRANCH
003424  101207  040000                  BIS     #DRINIT,R0        ; CLEAR INITIALIZE FLAG
003426  100657  000047                  MOV     R0,U.RCOV(R5)     ; SAVE RECOVERY STATUS
003430  104301  002172                  MOV     HIBN,R1           ; GET ORIGINAL STATE
003432  104167                  34$:    MOV     (SP),R0           ; RESTORE ERROR POINTER
003433  100471                          MOV     R1,-(R0)          ; SAVE STATE IN ERROR MESSAGE
003434  115000  002230                  TST     ERRPOS            ; SEE IF STATUS VALID
003436  073511                          BMI     26$               ; IF SO, BRANCH
003437                                  ASSUME  VALID,100000      ; ASSUME VALID BIT IS MSB
003437  020775                          CALL    RTDSL             ; GET REAL TIME DRIVE STATE
003440  115002                          TST     R2                ; SEE IF VALID
003441  053477                          BNE     24$               ; IF NOT, BRANCH
003442  102201  000101                  BIT     #RCVRDY!AVAIL,R1  ; SEE IF RECEIVER READY OR AVAILABLE
003444  013477                          BEQ     24$               ; IF NOT, BRANCH
003445  104204  001750                  MOV     #MAXSND,R4        ; SET UP TIMEOUT
003447  104203  001617                  MOV     #CR.GST,R3        ; POINT TO GET STATUS COMMAND
003451  104652  000025                  MOV     U.MASK(R5),R2     ; GET UNIT SDI SELECT MASK
003453  104137                  21$:    MOV     (R3),R0           ; POINTS TO SDI COMMAND BUFFER
003454                                  ASSUME  L2.OPC,0
003454  104631  000001                  MOV     L2.SLN(R3),R1     ; LOAD BYTE COUNT
003456  060004                          XFC     SEND              ; SEND SDI COMMAND
003457  115001                          TST     R1                ; SEE IF SDI COMMAND SENT SUCESSFULLY
003460  013464                          BEQ     22$               ; IF SO, BRANCH
003461  117404                          DEC     R4                ; DECREMENT TIMEOUT
003462  053453                          BNE     21$               ; IF TIMEOUT INCOMPLETE, BRANCH
003463  003477                          BR      24$               ; BRANCH
003464  104654  000033          22$:    MOV     U.SDIS(R5),R4     ; R4 HAS SHORT TIMEOUT
```

```
     56 003466  104207  001702          23$:    MOV     #ST,R0          ; POINT TO RECEIVE BUFFER
     57 003470  104631  000002                  MOV     L2.RLN(R3),R1   ; NUMBER OF WORDS IN RESPONSE
     58 003472  060005                          XFC     RCV             ; RECEIVE SDI RESPONSE
     59 003473  115001                          TST     R1              ; SEE IF SDI RESPONSE RECEIVED SUCESSFULLY
     60 003474  013511                          BEQ     26$             ; IF SO, BRANCH
     61 003475  117404                          DEC     R4              ; DECREMENT TIMEOUT VALUE
     62 003476  053466                          BNE     23$             ; IF TIMEOUT UNEXPIRED, BRANCH
     63 003477                          24$:    POP     <R0,R4>         ; RESTORE ERROR AND SUBUNIT POINTER
                                                                                        MOV (SP)+,R0
        003477  104267                                                                  MOV (SP)+,R4
        003500  104264
     64 003501  104203  177777                  MOV     #-1,R3          ; R3 CONTAINS 'UNABLE TO GET STATUS'
     65 003503  104202  000007                  MOV     #7,R2           ; R2 CONTAINS COUNT
     66 003505  100273                  25$:    MOV     R3,(R0)+        ; MOVE TO OUTPUT BUFFER
     67 003506  117402                          DEC     R2              ; DECREMENT COUNT
     68 003507  053505                          BNE     25$             ; IF COUNT INCOMPLETE, BRANCH
     69 003510  003540                          BR      8$              ; BRANCH
     70 003511                          26$:    POP     <R0,R4>         ; RESTORE ERROR AND SUBUNIT POINTER
                                                                                        MOV (SP)+,R0
        003511  104267                                                                  MOV (SP)+,R4
        003512  104264
     71 003513  104201  001711                  MOV     #ST+7,R1        ; R1 POINTS TO AFTER STATUS BUFFER
     72 003515  104202  000007                  MOV     #7,R2           ; R2 HAS COUNT
     73 003517  104413                  27$:    MOV     -(R1),R3        ; MOVE STATUS WORD TO R3
     74 003520  100273                          MOV     R3,(R0)+        ; MOVE TO OUTPUT BUFFER
     75 003521  117402                          DEC     R2              ; DECREMENT COUNT
     76 003522  053517                          BNE     27$             ; BRANCH IF COUNT INCOMPLETE
     77 003523  102200  000002  002226  20$:    BIT     #ATTN,ERMODE    ; SEE IF UNEXPECTED ATTENTION
     78 003526  013540                          BEQ     8$              ; IF NOT, BRANCH
     79                                 ;       SEE IF ERROR WAS CAUSE OF ATTENTION
     80 003527  104301  001704                  MOV     ST+ST.ERR,R1    ; GET ERROR BYTE
     81 003531  103201  177407                  BIC     #<HIBYTE!7>,R1  ; CLEAR UNUSED BITS
     82 003533  053540                          BNE     8$              ; IF ERROR(S), BRANCH
     83                                 ;       SEE IF LOGGABLE INFORMATION IS CAUSE OF ATTENTION
     84 003534  102200  000010  001703          BIT     #ST.EL,ST+ST.REQ        ; SEE IF LOGGABLE INFO
     85 003537  013605                          BEQ     50$             ; IF NOT, BRANCH ('RECOVER')
```

```
      1 003540  104650  000063  001106  8$:     MOV     U.UNUM(R5),HRQ.03       ; GET BASE NUMBER
      2 003543  105650  000050  001106          ADD     U.SUBU(R5),HRQ.03       ; ADD SUBUNIT OFFSET
      3 003546  104307  001103                  MOV     HRQ.RQ,R0       ; MOVE REQUEST TO R0
      4 003550  021053                          CALL    HOSTRQ          ; REPORT TO HOST
      5 003551  115000  001104                  TST     HRQ.01          ; SEE IF UNIT DROPPED
      6 003553  073636                          BMI     80$             ; IF SO, BRANCH
      7 003554                                  ASSUME  DROP,100000     ; HOST DROP BIT = BIT 15
      8 003554  104652  000047                  MOV     U.RCOV(R5),R2   ; SEE IF RECOVERY IS ENABLED
      9 003556  073671                          BMI     90$             ; IF NOT, BRANCH
     10 003557                                  ASSUME  ERMASK,100000   ; ASSUME RECOVERY MASK IS SIGN BIT
     11 003557  102200  000002  002226          BIT     #ATTN,ERMODE    ; DID WE HAVE AN ERROR?
     12 003562  013605                          BEQ     50$             ; IF NOT, BRANCH
     13 003563                                  REPSFT  SOFT            ; ELSE, REPORT SOFT ERROR
        003563  104200  000001  001105                                          MOV     #1,HRQ.02
        003566  114000  001106                                                  CLR     HRQ.03
        003570  114000  001107                                                  CLR     HRQ.04
        003572  100467                                                                  MOV R0,-(SP)
        003573  104657  000063                                                  MOV     U.UNUM(R5),R0
        003575  105657  000050                                                  ADD     U.SUBU(R5),R0
        003577  104070  001104                                                  MOV     R0,HRQ.01
        003601  104207  060007                                                  MOV     #T4SOFT,R0
        003603  021053                                                          CALL    HOSTRQ
        003604  104267                                                                  MOV (SP)+,R0
     14 003605  023700                  50$:    CALL    RECOVR          ; RECOVER FROM ERRORS
     15 003606  115002                          TST     R2              ; SEE IF ERRORS OCCURRED
     16 003607  053352                          BNE     40$             ; IF SO, LOOP
     17 003610                                  POP     <R0,R1>         ; RESTORE R0, R1
        003610  104267                                                                  MOV (SP)+,R0
        003611  104261                                                                  MOV (SP)+,R1
     18 003612  104653  000047                  MOV     U.RCOV(R5),R3   ; GET RECOVERY WORD
     19 003614  104032                          MOV     R3,R2           ; STORE IN R2
     20 003615  103202  001000                  BIC     #RETRY,R2       ; CLEAR RETRY BIT
     21 003617  100652  000047                  MOV     R2,U.RCOV(R5)   ; STORE RECOVERY WORD
     22 003621  102203  006000                  BIT     #SEKREQ!RCBREQ,R3       ; SEE IF SEEK NEEDED
     23 003623  013627                          BEQ     9$              ; IF NOT, BRANCH
     24 003624  104207  000023                  MOV     #SEEK,R0        ; SEEK NEXT
     25 003626  003675                          BR      NOSUB           ; EXIT
     26 003627  102203  001000          9$:     BIT     #RETRY,R3       ; SEE IF LAST MODULE SHOULD BE RETRIED
     27 003631  013673                          BEQ     100$            ; IF NOT, BRANCH
     28 003632  104307  002222                  MOV     LSTOVL,R0       ; RETRY LAST OVERLAY
     29 003634  114001                          CLR     R1              ; IMMIDATELY
     30 003635  003673                          BR      100$            ; EXIT
     31 003636                          80$:    POP     <R0,R1>         ; RESTORE REGISTERS
        003636  104267                                                                  MOV (SP)+,R0
        003637  104261                                                                  MOV (SP)+,R1
     32 003640  024172                          CALL    DSABLE          ; DISABLE ERROR RECOVERY
     33 003641  115000  002226                  TST     ERMODE          ; SEE IF ENTIRE UNIT DROPPED
     34 003643  033654                          BPL     10$             ; IF NOT, BRANCH
     35 003644                                  ASSUME  DROP,100000     ; ASSUME DROP SIGN BIT
     36 003644  114001                          CLR     R1              ; IMMIDATE CALL
     37 003645  104207  000026                  MOV     #DRPALL,R0      ; DRPALL NEXT
     38 003647  102200  000001  002223          BIT     #INTINP,M.PARM  ; SEE IF INITIALIZATION IN PROGRESS
     39 003652  013673                          BEQ     100$            ; IF NOT, BRANCH
     40 003653  003675                          BR      NOSUB           ; IF SO, JUST RETURN TO TROOT
     41 003654  104142                  10$:    MOV     (R4),R2         ; GET SUBUNIT PARAMETERS
     42 003655                                  ASSUME  S.PARM,0
     43 003655  101202  100000                  BIS     #DROP,R2        ; DROP SUBUNIT
```

```
     44 003657  100142                          MOV     R2,(R4)          ; SAVE
     45 003660                                  ASSUME  S.PARM,0
     46 003660  104651  000046                  MOV     U.PARM(R5),R1    ; GET UNIT PARAMETERS
     47 003662  101201  001000                  BIS     #FTIME,R1        ; MARK AS FIRST TIME
     48 003664  100651  000046                  MOV     R1,U.PARM(R5)    ; SAVE
     49 003666  104207  000001                  MOV     #SETUP,R0        ; SETUP NEXT MODULE
     50 003670  003675                  ..      BR      NOSUB            ; EXIT
     51 003671                          90$:    POP     <R0,R1>          ; RESTORE REGISTERS
        003671  104267                                                                  MOV (SP)+,R0
        003672  104261                                                                  MOV (SP)+,R1
     52 003673  115001                  100$:   TST     R1               ; SEE IF IMMIDATE CALL
     53 003674  012601                          BEQ     SEQLP            ; IF SO, BRANCH
     54 003675  100657  000013          NOSUB:  MOV     R0,U.NFUN(R5)    ; SAVE NEXT FUNCTION
     55 003677  000000                          RETURN                   ; RETURN TO ROOT
     56
```

```
                                        .SBTTL  RECOVR - RECOVER FROM THE ERROR
003700                                  RECOVR:
                                        ;
                                        ;       SET UP DRIVE AS IT SHOULD BE
                                        ;
003700  020746                                 CALL    RTDS            ; SEE IF WE CAN GET VALID STATE
003701  115002                                 TST     R2              ; SEE IF WE HAVE VALID STATE
003702  054061                                 BNE     10$             ; IF NOT, BRANCH
003703  104203  001612                        MOV     #CR.ONL,R3      ; POINT TO ONLINE COMMAND
003705  021173                                 CALL    TALK            ; INITIATE SDI INTERCHANGE
003706  115002                                 TST     R2              ; SEE IF ERROR OCCURRED
003707  013714                                 BEQ     1$              ; IF NOT, BRANCH
003710                                         CERROR  5,#SER2         ; REPORT SECONDARY ERROR
003710  104200  004665  001110                                                 MOV     #SER2,HRQ.05
003713  004061                                 BR      10$             ; BRANCH TO ERROR LOOP
003714  104203  001617                1$:     MOV     #CR.GST,R3      ; POINT TO GET STATUS
003716  021173                                 CALL    TALK            ; SEND SDI INTERCHANGE
003717  115002                                 TST     R2              ; SEE IF AN ERROR OCCURRED
003720  013725                                 BEQ     2$              ; BRANCH IF NO ERROR
003721                                         CERROR  5,#SER0         ; REPORT SECONDARY ERROR
003721  104200  004630  001110                                                 MOV     #SER0,HRQ.05
003724  004061                                 BR      10$             ; LOOP AND TRY AGAIN
003725  104207  177777                2$:     MOV     #-1,R0          ; SET UP FOR COMPLEMENT OF STATUS
003727  103307  001703                        BIC     ST+ST.REQ,R0    ; R0 HAS STATUS
003731  102207  000023                        BIT     #ST.PS!ST.SR!ST.RU,R0   ; SEE IF ANY FATAL STATES EXIST
003733  014004                                 BEQ     6$              ; IF NOT, BRANCH
003734                                         DEVFTL  23              ; REPORT SWITCH OUT OR SPINDLE NOT READY
003734  104200  002157  001107                                                 MOV     #ER23,HRQ.04
003737  104202  047667                                                         MOV     #23!FTLDEV+4000.,R2
003741  104020  001105                                                         MOV     R2,HRQ.02
003743  104200  003743  001104                                                 MOV     #.,HRQ.01
003746  104200  060014  001103                                                 MOV     #ERRMC,HRQ.RQ
003751  102207  000001                        BIT     #ST.RU,R0       ; SEE IF RUN/STOP SWITCH OUT
003753  013760                                 BEQ     3$              ; IF NOT, BRANCH
003754                                         CERROR  5,#SER42        ; REPORT RUN SWITCH OUT
003754  104200  002200  001110                                                 MOV     #SER42,HRQ.05
003757  003772                                 BR      5$              ; BRANCH
003760  102207  000020                3$:     BIT     #ST.SR,R0       ; SEE IF SPINDLE DROPPED READY
003762  013767                                 BEQ     4$              ; IF NOT, BRANCH
003763                                         CERROR  5,#SER43        ; REPORT SPINDLE NOT READY
003763  104200  002214  001110                                                 MOV     #SER43,HRQ.05
003766  003772                                 BR      5$              ; BRANCH
003767                                 4$:     CERROR  5,#SER44        ; REPORT PORT SWITCH OUT
003767  104200  002231  001110                                                 MOV     #SER44,HRQ.05
003772                                 5$:     ENDERR
003772  104200  000051  001111                                                 MOV     #SER22,HRQ.06
003775  104200  000007  002230                                                 MOV     #6+1,ERRPOS     ; SET THE POSITION
004000  101200  100000  002230                BIS     #100000,ERRPOS  ; STATUS VALID
004003  004061                                 BR      10$             ; BRANCH
004004  102207  000100                6$:     BIT     #ST.RR,R0       ; SEE IF DRIVE IS TO BE RECALIBRATED
004006  054010                                 BNE     7$              ; IF NOT, BRANCH (NOTE COMPLEMENT)
004007  024215                                 CALL    GORCLB          ; MARK AS RECALIBRATION NEEDED
004010  104307  001703                7$:     MOV     ST+ST.MOD,R0    ; GET MODE BITS
004012  110707                                 SWAB    R0              ; MOVE WRITE PROTECT BUTTON STATUS TO LO BYTE
004013  103207  177417                        BIC     #LBHINB,R0      ; CLEAR UNUSED BITS
004015  100657  000026                        MOV     R0,U.WRIT(R5)   ; SAVE WRITE PROTECT STATUS FOR CHANGE MODE
004017  104301  001704                        MOV     ST+ST.ERR,R1    ; GET ERROR BITS
```

RECOVR - RECOVER FROM THE ERROR

```
     46 004021  103201  177400                BIC     #HIBYTE,R1      ; CLEAR UNUSED BITS
     47 004023  014043                      BEQ     9$              ; IF NO ERRORS, BRANCH
     48 004024  102201  000200              BIT     #ST.DE,R1       ; SEE IF A DRIVE ERROR OCCURRED
     49 004026  014030                      BEQ     11$             ; IF NOT, BRANCH
     50 004027  024207                      CALL    GOSEEK          ; SEEK IS REQUIRED
     51 004030  104010  001667        11$:  MOV     R1,DCLR         ; MOVE ERROR BITS TO CLEAR DRIVE COMMAND
     52 004032  104203  001624              MOV     #CR.CLR,R3      ; POINT TO SDI COMMAND
     53 004034  021173                      CALL    TALK            ; INITIATE SDI INTERCHANGE
     54 004035  115002                      TST     R2              ; SEE IF ANY ERRORS OCCURRED
     55 004036  014043                      BEQ     9$              ; IF NOT, BRANCH
     56 004037                              CERROR  5,#SER1         ; REPORT SECONDARY ERROR
        004037  104200  004646  001110                                      MOV     #SER1,HRQ.05
     57 004042  004061                      BR      10$             ; BRANCH TO ERROR LOOP
     58 004043  104653  000026        9$:   MOV     U.WRIT(R5),R3   ; GET WRITE PROTECTION BUTTON STATUS
     59 004045  101653  000045              BIS     U.WPRT(R5),R3   ; SET WRITE PROT BITS FOR READ ONLY DRIVES
     60 004047  101203  177402              BIS     #177402,R3      ; SET OTHER CHANGE MODE BITS
     61 004051  104030  001665              MOV     R3,WRITBT       ; MOVE TO CHANGE MODE COMMAND
     62 004053  104203  001631              MOV     #CR.MOD,R3      ; POINT TO MODE COMMAND
     63 004055  021173                      CALL    TALK            ; INITIATE SDI INTERCHANGE
     64 004056                              CERROR  5,#SER3         ; REPORT SECONDARY ERROR (IF NO ERROR, DON'T CARE)
        004056  104200  004707  001110                                      MOV     #SER3,HRQ.05
     65 004061  114000  002226        10$:  CLR     ERMODE          ; RESET ERMODE
     66 004063  000000                      RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  CHKDN  - FIND PREVIOUS ACTIVE SUBUNIT
004064                                  CHKDN:
                                        ;
                                        ;       SEARCH THE SUBUNIT LIST DOWN AND TRY TO FIND AN ACTIVE SUBUNIT
                                        ;
004064  104203  000001                          MOV     #U.SUBP,R3      ; R3 WILLPOINT TO SUBUNIT POINTERS
004066  105053                                  ADD     R5,R3           ; R3 POINTS TO SUBUNIT POINTERS
004067  117402                          1$:     DEC     R2              ; DECREMENT OLD SUBUNIT
004070  074104                                  BMI     2$              ; IF ALL SUBUNITS CHECKED, BRANCH
004071  104034                                  MOV     R3,R4           ; MOVE SUBUNIT LIST POINTER TO R4
004072  105024                                  ADD     R2,R4           ; POINT TO SUBUNIT
004073  104144                                  MOV     (R4),R4         ; R4 POINTS TO SUBUNIT
004074  074067                                  BMI     1$              ; IF NO UNIT, BRANCH
004075  104147                                  MOV     (R4),R0         ; R0 HAS SUBUNIT PARAMETERS
004073                                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
004076  074067                                  BMI     1$              ; IF THIS SUBUNIT DROPPED, BRANCH
004077                                          ASSUME  DROP,100000     ; ASSUME DROP IS SIGN BIT
004077  100652  000050                          MOV     R2,U.SUBU(R5)   ; SAVE SUBUNIT NUMBER
004101  114007                                  CLR     R0              ; FOUND A SUBUNIT
004102  114003                                  CLR     R3              ; PROCESS THIS SUBUNIT
004103  004113                                  BR      3$              ; EXIT
004104  104657  000046          2$:     MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
004106  103207  010000                  BIC     #DIREC,R0       ; SET DIRECTION UP
004110  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE
004112  104057                          MOV     R5,R0           ; DIDN'T FIND A SUBUNIT
004113  000000          3$:     RETURN                          ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  CHKDN  - FIND FOLLOWING ACTIVE SUBUNIT
004114                                  CHKUP:
                                        ;
                                        ;       SEARCH THE SUBUNIT LIST UP AND TRY TO FIND AN ACTIVE SUBUNIT
                                        ;
004114  104203  000001                          MOV     #U.SUBP,R3      ; R3 WILLPOINT TO SUBUNIT POINTERS
004116  105053                                  ADD     R5,R3           ; R3 POINTS TO SUBUNIT POINTERS
004117  115402                          1$:     INC     R2              ; INCREMENT OLD SUBUNIT
004120  106202  000003                          CMP     #3,R2           ; SEE IF ALL SUBUNITS CHECKED
004122  074144                                  BMI     3$              ; IF ALL SUBUNITS CHECKED, BRANCH
004123  104034                                  MOV     R3,R4           ; MOVE SUBUNIT LIST POINTER TO R4
004124  105024                                  ADD     R2,R4           ; POINT TO SUBUNIT
004125  104144                                  MOV     (R4),R4         ; R4 POINTS TO SUBUNIT
004126  074117                                  BMI     1$              ; IF NO UNIT, BRANCH
004127  104147                                  MOV     (R4),R0         ; R0 HAS SUBUNIT PARAMETERS
004130                                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
004130  074117                                  BMI     1$              ; IF THIS SUBUNIT DROPPED, BRANC
004131                                          ASSUME  DROP,100000     ; ASSUME DROP IS SIGN BIT
004131  115000  002223                          TST     M.PARM          ; SEE IF INITIAL WRITE IN PROGRESS
004133  034137                                  BPL     2$              ; IF NOT, BRANCH
004134                                          ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
004134  102207  040000                          BIT     #INITW,R0       ; SEE IF INITIAL WRITE TO BE DONE ON SUBUNIT
004136  014117                                  BEQ     1$              ; IF NOT, BRANCH
004137  100652  000050                  2$:     MOV     R2,U.SUBU(R5)   ; SAVE SUBUNIT NUMBER
004141  114007                                  CLR     R0              ; FOUND ONE
004142  114003                                  CLR     R3              ; PROCESS THIS ONE
004143  004162                                  BR      5$              ; EXIT
004144  104651  000046                  3$:     MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
004146  115000  002223                          TST     M.PARM          ; SEE IF DOING AN INITIAL WRITE
004150  034156                                  BPL     4$              ; IF NOT, BRANCH
004151                                          ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
004151  103201  040000                          BIC     #INITW,R1       ; FLAG INITIAL WRITE AS COMPLETE ON THIS UNIT
004153  101201  001000                          BIS     #FTIME,R1       ; START TESTING UNIT
004155  004160                                  BR      9$              ; EXIT
004156  101201  010000                  4$:     BIS     #DIREC,R1       ; SET DIRECTON DOWN
004160  100651  000046                  9$:     MOV     R1,U.PARM(R5)   ; SAVE
004162  000000                          5$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  ERROR RECOVERY SUPPORT SUBROUTINES
                                .SBTTL  ENABLE - ENABLE ERROR RECOVERY
004163                          ENABLE:
                                ;
                                ;       ENABLE ERROR RECOVERY
                                ;
004163  104653  000047                  MOV     U.RCOV(R5),R3   ; GET RECOVERY WORD
004165  103203  100000                  BIC     #ERMASK,R3      ; CLEAR MASK
004167  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
004171  000000                          RETURN                  ; RETURN
                                .SBTTL  DSABLE - DISABLE ERROR RECOVERY
004172                          DSABLE:
                                ;
                                ;       DISABLE ERROR RECOVERY
                                ;
004172  104653  000047                  MOV     U.RCOV(R5),R3   ; GET RECOVERY WORD
004174  101203  100000                  BIS     #ERMASK,R3      ; SET MASK
004176  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
004200  000000                          RETURN                  ; RETURN
                                .SBTTL  GORTRY - RETRY OF MODULE REQUIRED TO RECOVER
004201                          GORTRY:
                                ;
                                ;       JUST RETRY THE LAST MODULE TO RECOVER
                                ;
004201  024163                          CALL    ENABLE          ; ENABLE ERROR RECOVERY
004202  101203  001000                  BIS     #RETRY,R3       ; SET RETRY BIT
004204  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
004206  000000                          RETURN                  ; RETURN
                                .SBTTL  GOSEEK - SEEK REQUIRED TO RECOVER
004207                          GOSEEK:
                                ;
                                ;       GOSEEK WILL FLAG THAT THE DRIVE MUST SEEK TO RECOVER FROM AN ERROR
                                ;
004207  024163                          CALL    ENABLE          ; ENABLE ERROR RECOVERY
004210  101203  004000                  BIS     #SEKREQ,R3      ; SET SEEK REQUIRED BIT
004212  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
004214  000000                          RETURN                  ; RETURN
                                .SBTTL  GORCLB - RECALIBRATE REQUIRED TO RECOVER
004215                          GORCLB:
                                ;
                                ;       GORCLB WILL FLAG THAT THE DRIVE MUST RECALIBRATE TO RECOVER FROM AN ERROR
                                ;
004215  024163                          CALL    ENABLE          ; ENABLE ERROR RECOVERY
004216  101203  006000                  BIS     #SEKREQ!RCBREQ,R3       ; SET SEEK AND RECALIBRATE PEQUIRED BIT
004220  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
004222  000000                          RETURN                  ; RETURN
                                .SBTTL  GOINIT - DRIVE MUST BE INITIALIZED TO RECOVER
004223                          GOINIT:
                                ;
                                ;       INIT THE DRIVE, THEN FLAG AS DONE
                                ;
004223                                  PUSH    R1              ; SAVE R1 (DON'T KNOW IF NEEDED)
004223  100461                                                          MOV R1,-(SP)
004224  020775                          CALL    RTDSL           ; GET DRIVE STATE BEFORE INIT
004225  104010  002172                  MOV     R1,HIBN         ; SAVE IN UNUSED LOCATION
004227  104652  000025                  MOV     U.MASK(R5),R2   ; GET PORT MASK
004231  060011                          XFC     DINIT           ; INIT THE DRIVE
```

```
     57 004232  024207                        CALL    GOSEEK          ; SEEK REQUIRED IF INITED
     58 004233  101203  040000                BIS     #DRINIT,R3      ; MARK THE DRIVE AS INITIALIZED
     59 004235  100653  000047                MOV     R3,U.RCOV(R5)   ; SAVE
     60 004237  104201  001400                MOV     #1400,R1        ; MOVE TIMEOUT TO R1
     61 004241  117401                  1$:   DEC     R1              ; DECREMENT TIMEOUT (WAIT FOR DRIVE CLOCKS)
     62 004242  054241                        BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
     63 004243                                POP     R1              ; RESTORE R1
        004243  104261                                                                  MOV (SP)+,R1
     64 004244  000000                        RETURN                  ; RETURN
```

```
                                .SBTTL  OVERLAY DATA STRUCTURES AND ASSEMBLY ERROR CHECKING
                                ;
                                ;       THE PARTITION TABLE HAS THE PARTITION START ADDRESSES AS WELL
                                ;       AS HOLDING THE NUMBER OF THE OVERLAY LOADED CURRENTLY (IF NO
                                ;       OVERLAY IS LOADED, THE OVERLAY NUMBER IS -1
                                ;
 004245                         PTABLE:
 004245  177777                 PART0:  .WORD   -1
 004246  004413                         .WORD   AREA0
 004247  177777                 PART1:  .WORD   -1
 004250  004602                         .WORD   AREA1
 004251  177777                 PART2:  .WORD   -1
 004252  005043                         .WORD   AREA2
 004253  177777                 PART3:  .WORD   -1
 004254  005230                         .WORD   BUFARA          ; PARTITION 3 GOES IN READ/WRITE BUFFER AREA
 004255  177777                 PARTI:  .WORD   -1
 004256  004701                         .WORD   AREAI
                                ;
                                ;       EACH ENTRY IN THE OVERLAY TABLE IS COMPOSED OF FOUR WORDS:
                                ;
                                ;       0)      PARTITION TO LOAD OVERLAY INTO
                                ;       1)      NUMBER OF FOLLOWING PARTITIONS THAT ARE DISTROYED
                                ;       2)      LOW ORDER STARTING UNIBUS ADDRESS OF OVERLAY
                                ;       3)      <1::0> HI ORDER STARTING UNIBUS ADDRES
                                ;               <15::2> LENGTH OF OVERLAY
                                ;
 000000                         MSSG$   =       0
 000000                         OCNT$   =       0
 004257                         OTABLE: DFOVLY  MSSG$,MS,PART0          ; MESSAGES
 004257  000000                         .WORD   <PART0-PART0>/2
 004260  000000                         .WORD   0
 004261  000000                         .WORD   0
 004262  027574                         .WORD   OVL.MS*4
 004263                                 DFOVLY  SETUP,SU,PART0,PART3    ; SETUP
 004263  000000                         .WORD   <PART0-PART0>/2
 004264  000003                         .WORD   <PART3-PART0>/2
 004265  000000                         .WORD   0
 004266  000734                         .WORD   OVL.SU*4
 004267                                 DFOVLY  NEWOP,NO,PART1          ; NEWOP
 004267  000001                         .WORD   <PART1-PART0>/2
 004270  000000                         .WORD   0
 004271  000000                         .WORD   0
 004272  001204                         .WORD   OVL.NO*4
 004273                                 DFOVLY  RNDBE,RB,PART2          ; RNDBE
 004273  000002                         .WORD   <PART2-PART0>/2
 004274  000000                         .WORD   0
 004275  000000                         .WORD   0
 004276  000700                         .WORD   OVL.RB*4
 004277                                 DFOVLY  SEQBE,SB,PART2          ; SEQBE
 004277  000002                         .WORD   <PART2-PART0>/2
 004300  000000                         .WORD   0
 004301  000000                         .WORD   0
 004302  002574                         .WORD   OVL.SB*4
 004303                                 DFOVLY  RNDTG,RT,PART2          ; RNDTG
 004303  000002                         .WORD   <PART2-PART0>/2
 004304  000000                         .WORD   0
```

```
                004305  000000          .WORD   0
                004306  000660          .WORD   OVL.RT*4
             36 004307                  DFOVLY  SEQTG,ST,PART2          ; SEQTG
                004307  000002          .WORD   <PART2-PART0>/2
                004310  000000          .WORD   0
                004311  000000          .WORD   0
                004312  002344          .WORD   OVL.ST*4
             37 004313                  DFOVLY  BUILDP,BP,PART0,PART3   ; BUILDP
                004313  000000          .WORD   <PART0-PART0>/2
                004314  000003          .WORD   <PART3-PART0>/2
                004315  000000          .WORD   0
                004316  001344          .WORD   OVL.BP*4
             38 004317                  DFOVLY  WRITE,W,PART0           ; WRITE
                004317  000000          .WORD   <PART0-PART0>/2
                004320  000000          .WORD   0
                004321  000000          .WORD   0
                004322  002020          .WORD   OVL.W*4
             39 004323                  DFOVLY  AFTWRT,AW,PART0         ; AFTWRT
                004323  000000          .WORD   <PART0-PART0>/2
                004324  000000          .WORD   0
                004325  000000          .WORD   0
                004326  003020          .WORD   OVL.AW*4
             40 004327                  DFOVLY  READ,R,PART0            ; READ
                004327  000000          .WORD   <PART0-PART0>/2
                004330  000000          .WORD   0
                004331  000000          .WORD   0
                004332  001330          .WORD   OVL.R*4
             41 004333                  DFOVLY  SECCHK,SC,PART0         ; SECCHK
                004333  000000          .WORD   <PART0-PART0>/2
                004334  000000          .WORD   0
                004335  000000          .WORD   0
                004336  003064          .WORD   OVL.SC*4
             42 004337                  DFOVLY  CHKEDC,ED,PART0         ; CHKEDC
                004337  000000          .WORD   <PART0-PART0>/2
                004340  000000          .WORD   0
                004341  000000          .WORD   0
                004342  001734          .WORD   OVL.ED*4
             43 004343                  DFOVLY  CHKECC,CK,PART0         ; CHKECC
                004343  000000          .WORD   <PART0-PART0>/2
                004344  000000          .WORD   0
                004345  000000          .WORD   0
                004346  002074          .WORD   OVL.CK*4
             44 004347                  DFOVLY  ERCOV,EC,PART0          ; ERCOV
                004347  000000          .WORD   <PART0-PART0>/2
                004350  000000          .WORD   0
                004351  000000          .WORD   0
                004352  000360          .WORD   OVL.EC*4
             45 004353                  DFOVLY  NEWLEV,NL,PART0         ; NEWLEV
                004353  000000          .WORD   <PART0-PART0>/2
                004354  000000          .WORD   0
                004355  000000          .WORD   0
                004356  000704          .WORD   OVL.NL*4
             46 004357                  DFOVLY  CMPDAT,CD,PART0         ; CMPDAT
                004357  000000          .WORD   <PART0-PART0>/2
                004360  000000          .WORD   0
                004361  000000          .WORD   0
                004362  001404          .WORD   OVL.CD*4
```

```
     47 004363                                  DFOVLY  LSTMOD,LM,PART0          ; LSTMOD
        004363  000000                          .WORD   <PART0-PART0>/2
        004364  000000                          .WORD   0
        004365  000000                          .WORD   0
        004366  000270                          .WORD   OVL.LM*4
     48 004367                                  DFOVLY  REVCT,RV,PART0           ; REVCT
        004367  000000                          .WORD   <PART0-PART0>/2
        004370  000000                          .WORD   0
        004371  000000                          .WORD   0
        004372  002640                          .WORD   OVL.RV*4
     49 004373                                  DFOVLY  SEEK,SK,PART3            ; SEEK
        004373  000003                          .WORD   <PART3-PART0>/2
        004374  000000                          .WORD   0
        004375  000000                          .WORD   0
        004376  001514                          .WORD   OVL.SK*4
     50 004377                                  DFOVLY  SEKTST,TS,PART3          ; SEKTST
        004377  000003                          .WORD   <PART3-PART0>/2
        004400  000000                          .WORD   0
        004401  000000                          .WORD   0
        004402  001744                          .WORD   OVL.TS*4
     51 004403                                  DFOVLY  RECALB,RC,PART3          ; RECALB
        004403  000003                          .WORD   <PART3-PART0>/2
        004404  000000                          .WORD   0
        004405  000000                          .WORD   0
        004406  000300                          .WORD   OVL.RC*4
     52 004407                                  DFOVLY  DRPALL,DA,PART0,PART3    ; DRPALL
        004407  000000                          .WORD   <PART0-PART0>/2
        004410  000003                          .WORD   <PART3-PART0>/2
        004411  000000                          .WORD   0
        004412  000254                          .WORD   OVL.DA*4
     53                                 ;
     54                                 ;       EVERYTHING FOLLOWING THE AREA0 DEFINITION IS IN OVERLAY PARTITION
     55                                 ;       SPACE AND WILL BE DESTROYED AFTER INITIALIZATION
     56                                 ;
     57              004413             AREA0   =       .
     58 004413                                  DFOVLY  INSET,IN,PART1           ; INSET
        004413  000004                          .WORD   <PART1-PART0>/2
        004414  000000                          .WORD   0
        004415  000000                          .WORD   0
        004416  000110                          .WORD   OVL.IN*4
     59 004417                                  DFOVLY  COMCHR,CC,PART1          ; COMCHR
        004417  000004                          .WORD   <PART1-PART0>/2
        004420  000000                          .WORD   0
        004421  000000                          .WORD   0
        004422  000614                          .WORD   OVL.CC*4
     60 004423                                  DFOVLY  SPINUP,SP,PART1          ; SPINUP
        004423  000004                          .WORD   <PART1-PART0>/2
        004424  000000                          .WORD   0
        004425  000000                          .WORD   0
        004426  000134                          .WORD   OVL.SP*4
     61 004427                                  DFOVLY  SORT,SO,PART1            ; SORT
        004427  000004                          .WORD   <PART1-PART0>/2
        004430  000C00                          .WORD   0
        004431  000000                          .WORD   0
        004432  001110                          .WORD   OVL.SO*4
     62 004433                                  DFOVLY  SCHAR0,SO,PART1          ; SCHAR0
        004433  000004                          .WORD   <PART1-PART0>/2
```

```
                004434  000000                          .WORD   0
                004435  000000                          .WORD   0
                004436  004404                          .WORD   OVL.S0*4
            63  004437                                  DFOVLY  SCHAR1,S1,PARTI         ; SCHAR1
                004437  000004                          .WORD   <PARTI-PART0>/2
                004440  000000                          .WORD   0
                004441  000000                          .WORD   0
                004442  003120                          .WORD   OVL.S1*4
            64  004443                                  DFOVLY  GETSER,GS,PARTI         ; GETSER
                004443  000004                          .WORD   <PARTI-PART0>/2
                004444  000000                          .WORD   0
                004445  000000                          .WORD   0
                004446  002250                          .WORD   OVL.GS*4
            65  004447                                  DFOVLY  INITD,ID,PARTI          ; INITD
                004447  000004                          .WORD   <PARTI-PART0>/2
                004450  000000                          .WORD   0
                004451  000000                          .WORD   0
                004452  000604                          .WORD   OVL.ID*4
            66          000037              NUMOVL      =       <. - OTABLE> / 4
            67  004453                                  ONETIM
                                                        .LIST   ME
                                            .SBTTL      UNIT AND SUBUNIT GET CHARACTERISTICS AND RUN SDI COMMANDS
                                            ;
                                            ;           NOTE: BECAUSE OF THEIR POSITION, THE FOLLOWING SDI LEVEL 2 COMMANDS
                                            ;           WILL BE ISSUED WITH A LONG TIMEOUT
                                            ;
                        000000              SDI$        =       0                   ; DOUBLE UP ON THESE COMMANDS, SINCE IF
                                                                                    ; AN ERROR OCCURS, IT'S A DEVICE FATAL
                004453                      CR.GCR:     MSG GCR,1,11.,CHRRES        ; GET CHARACTERISTICS
                004453  004473                          .WORD   GCR                         ; ADDRESS OF COMMAND
                004454  000001                          .WORD   1                           ; SIZE OF COMMAND IN BYTES
                004455  000013                          .WORD   11.                         ; SIZE OF REPLY IN WORDS
                004456  000170                          .WORD   CHRRES                      ; SUCCESSFUL COMPLETION CODE
                004457  000035                          .WORD   SDI$+U.SDI2                 ; RETRY COUNT OFFSET
                        000001              SDI$        =       SDI$+1
                004460                      CR.SCR:     MSG SCR,2,19.,SBCRES        ; GET SUBUNIT CHARACTERISTICS
                004460  004474                          .WORD   SCR                         ; ADDRFSS OF COMMAND
                004461  000002                          .WORD   2                           ; SIZE OF COMMAND IN BYTES
                004462  000023                          .WORD   19.                         ; SIZE OF REPLY IN WORDS
                004463  000167                          .WORD   SBCRES                      ; SUCCESSFUL COMPLETION CODE
                004464  000036                          .WORD   SDI$+U.SDI2                 ; RETRY COUNT OFFSET
                        000002              SDI$        =       SDI$+1
                004465                      CR.RUN:     MSG RUN,1,7,COMPLT          ; INITIATE LOAD
                004465  004472                          .WORD   RUN                         ; ADDRESS OF COMMAND
                004466  000001                          .WORD   1                           ; SIZE OF COMMAND IN BYTES
                004467  000007                          .WORD   7                           ; SIZE OF REPLY IN WORDS
                004470  000176                          .WORD   COMPLT                      ; SUCCESSFUL COMPLETION CODE
                004471  000037                          .WORD   SDI$+U.SDI2                 ; RETRY COUNT OFFSET
                        000003              SDI$        =       SDI$+1
                004472     000      014     RUN:        .BYTE   0,DRVRUN            ; DRIVE RUN
                004473     000      207     GCR:        .BYTE   0,GETCHR            ; GET CHARACTERISTICS WITH A
                004474     000      210     SCR:        .BYTE   0,GETSUB            ; GET SUBUNIT CHARACTERISTICS
                004475  000000              SUBUNT:     .WORD   0                   ; SUBUNIT SELECTION IN LOW ORDER BYTE
                                            ;
                                            ;           CHECK THAT 4 UNITS WITH 8 SUBUNITS IN ALL, WITH 16 BAD BLOCKS
                                            ;           SPECIFIED EACH, CAN FIT IN MEMORY WITHOUT OVERLAPPING WITH
                                            ;           THE OVERLAYS OR THE R/W BUFFER SPACE
```

```
                                ;
                                        .IF     DF,MAXADR
000010                          MAXSUB  =       8.
006232                          MINADR  =       HIMEM-<<4*<U.LGRP+1>>+<MAXSUB*<S.BESS+68.>>+2>
                                        .ENDC
                                        .IF     NDF,BUFARA
                                BUFARA  =       0                       ; TO DETERMINE READ/WRITE BUFFERS START
                                MAXADR  =       0                       ; TO DETERMINE MAXIMUM OVERLAY ADDRESS
                                        .ENDC
                                .SBTTL  ********** NON-OVERLAY MODULE START - TEST 4 INITIALIZATION
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;
                                ;       THIS 'OVERLAY' IS LOADED ALONG WITH THE INITIAL DOWNLINE LOAD OF
                                ;       TEST 4.  ONCE EXECUTED, THIS OVERLAY IS NEVER LOADED AGAIN UNLESS
                                ;       THE ENTIRE TEST IS STARTED (OR RESTARTED) AGAIN.  ALL OTHER
                                ;       OVERLAYS ARE LOADED INTO THE SPACE THAT THIS ONE OCCUPIES INITIALLY,
                                ;       THUS DESTROYING THIS OVERLAY.
                                ;
                                ;
004476                          UCURSR: .BLKW   1
004477  007702                  LASTU:  .WORD   FIRSTU
004500                          UMASK:  .BLKW   1
004501                          NUMBB:  .BLKW   1
004502                          MAXDBN: .BLKW   2
004504                          SECTRK: .BLKW   1
004505                          SECGRP: .BLKW   1
004506                          SECCYL: .BLKW   1
004507                          TGS:    .BLKW   7
004516                          DSERNM: .BLKW   3                       ; TEMP STORAGE FOR DRIVE SERIAL NUMBER
004521  024701                  START:  CALL    GMPARM                  ; GET MASTER PARAMETERS
004522  024755                          CALL    GETU                    ; GET ALL UNITS TO TEST
004523  025365                          CALL    BLDUNT                  ; BUILD UNIT AND SUBUNIT PARAMETERS
004524  101200  000001  002223         BIS     #INTINP,M.PARM          ; MARK AS INITIALIZATION IN PROGRESS
004527  024640                          CALL    TROOT                   ; FILL OUT UNIT AND SUBUNIT PARM'S, GET SUBUNIT CHAR
004530  102200  000002  002223         BIT     #DIE,M.PARM             ; SEE IF INITIALIZATION ERRORS
004533  014563                          BEQ     3$                      ; IF NOT, BRANCH
004534                                  DEVFTL  16,#SER39               ; REPORT INITIALIZATION ERRORS
004534                                  ERROR   FTLDEV,16,<#SER39>
        000012                          .RADIX  10
        000005                          NUMPTR  =       5
004534                                  MOVMSG  #ER16,4

                                        .IF     LT,4-10
004534  104200  001553  001107                                                  MOV     #ER16,HRQ.04
                                        .IFF
                                                                                MOV     #ER16,HRQ.4
                                        .ENDC
                                        .IF     NB,<#SER39>
                                        .IRP    X,<#SER39>
                                MOVMSG  X,\NUMPTR
                                NUMPTR  =       NUMPTR + 1
                                        .ENDR
004537                          MOVMSG  #SER39,\NUMPTR
```

```
                                        .IF     LT,5-10
004537  104200  002477  001110                                        MOV     #SER39,HRQ.05
                                        .IFF
                                                                      MOV     #SER39,HRQ.5
                                        .ENDC
        000006                          NUMPTR  =       NUMPTR + 1
                                        .ENDC
004542  104202  047660                                                MOV     #16!FTLDEV+4000.,R2
004544  104020  001105                                                MOV     R2,HRQ.02
004546  104200  004546  001104                                        MOV     #.,HRQ.01
        000010                          .RADIX
004551  104200  060014  001103                                        MOV     #ERRMC,HRQ.RQ
004554  104200  177777  001106        MOV     #-1,HRQ.03      ; MOVE 'NOT ASSICIATED WITH ANY UNIT' TO UNIT #
004557  104307  001103                MOV     HRQ.RQ,R0       ; GET REQUEST
004561  021053                      CALL    HOSTRQ          ; REPORT
004562  002573                      BR      STOP            ; STOP TEST 4
004563  103200  000001  002223  3$:  BIC     #INTINP,M.PARM  ; INITIALIZATION NO LONGER IN PROGRESS
004566  104307  002227              MOV     MEMPOL,R0       ; R0 HAS POINTER TO FREE MEMORY
004570  107207  005230              SUB     #BUFARA,R0      ; SUBTRACT END OF CODE FROM FREE MEMORY
004572  104073                      MOV     R0,R3           ; SAVE AMOUNT OF FREE MEMORY
004573  114001                      CLR     R1              ; CLEAR COUNT
004574  107207  000424          1$: SUB     #RBUFLN+LINKLN,R0       ; SUBTRACT NEEDED MEM FOR READ
004576  074601                      BMI     2$              ; IF NO MORE MEMORY, BRANCH
004577  115401                      INC     R1              ; INCREMENT COUNT
004600  004574                      BR      1$              ; LOOP
004601  104010  002233          2$: MOV     R1,SECMAX       ; SAVE IN SECTOR MAXIMUM
004603  114001                      CLR     R1              ; CLEAR COUNT
004604  107203  000410              SUB     #WBUFLN+LINKLN,R3       ; SUBTRACT NEEDED MEM FOR WRITE
004606  115401                  4$: INC     R1              ; INCREMENT COUNT
004607  107203  000007              SUB     #LINKLN,R3      ; SUBTRACT LINK LENGTH FROM MEMORY
004611  074613                      BMI     5$              ; IF MEMORY EXHAUSTED, BRANCH
004612  004606                      BR      4$              ; LOOP
004613  104010  002234          5$: MOV     R1,CHNMAX       ; SAVE IN SECTOR MAXIMUM
004615  002532                      BR      ROOT            ; START EXERCISE
                                    .SBTTL  GETMEM - ALLOCATES MEMORY FOR UNIT AND SUBUNIT BLOCK AND SUBUNIT PARAMETERS
004616                              GETMEM:
                                    ;
                                    ;       RETURN A BLOCK OF MEMORY
                                    ;
004616  107070  002227              SUB     R0,MEMPOL       ; SUBTRACT REQUESTED LENGTH FROM MEMOY POOL
004620  104307  002227              MOV     MEMPOL,R0       ; LOAD R0 WITH POINTER TO REQUESTED MEMORY
004622  000000                      RETURN                  ; RETURN TO CALLING PROGRAM
                                    .SBTTL  CBB2    - 28 BIT COMPARE FOR SETUP MODULES
004623                              CBB2:
                                    ;
                                    ;       28 BIT COMPARE FOR BAD BLOCKS
                                    ;
004623  104634  000001              MOV     1(R3),R4        ; GET WORD THAT R3 POINTS TO
004625  103204  170000              BIC     #^CHBHINB,R4    ; STRIP OFF UNUSED BITS
004627  104625  000001              MOV     1(R2),R5        ; GET OTHER WORD TO COMPARE
004631  103205  170000              BIC     #^CHBHINB,R5    ; STRIP OFF THE UNUSED BITS
004633  106045                      CMP     R4,R5           ; COMPARE
004634  054637                      BNE     1$              ; IF DIFFERENCE IS FOUND, BRANCH
004635  104125                      MOV     (R2),R5         ; GET OTHER WORD TO COMPARE
004636  106135                      CMP     (R3),R5         ; COMPARE
004637  000000                  1$: RETURN                  ; RETURN TO SORTBE
```

```
                                .SBTTL  TROOT  - TEMPORARY ROOT FOR SEQNCR DURING TEST 4 SETUP
004640                          TROOT:
                                ;
                                ;       TROOT WILL FILL OUT THE AS YET UNDEFINED FIELDS IN BOTH UNIT
                                ;       AND SUBUNIT PARAMETERS, AS WELL AS GETTING THE SUBUNIT CAHRACTERISTICS
                                ;
004640  104205  007702                MOV     #FIRSTU,R5      ; R5 POINTS TO FIRST SUBUNIT
004642  114002                  MKLOOP: CLR     R2              ; FOR ZEROING RECOVERY WORD
004643  100652  000047                MOV     R2,U.RCOV(R5)   ; ZERO RECOVERY WORD
004645  104207  001000                MOV     #FTIME,R0       ; MARK AS FIRST TIME (WILL DISABLE RECOVERY)
004647  100657  000046                MOV     R0,U.PARM(R5)   ; SAVE
004651  104201  000027                MOV     #INSET,R1       ; FIRST MODULE IS INSET
004653  100651  000013                MOV     R1,U.NFUN(R5)   ; SAVE IN UNIT PARAMETERS
004655  104201  000003                MOV     #3,R1           ; SDI SHORT TIMEOUT SET TO 30 SEC
004657  100651  000033                MOV     R1,U.SDIS(R5)   ; MOVE TO PARAMETERS
004661  104201  000360                MOV     #360,R1         ; WRITE PROTECT THE ENTIRE DRIVE
004663  100651  000045                MOV     R1,U.WPRT(R5)   ; SAVE
004665  022577                        CALL    SEQNCR          ; RUN SEQUENCER
004666  104657  000046                MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
004670  101207  001000                BIS     #FTIME,R0       ; SET FIRST TIME
004672  100657  000046                MOV     R0,U.PARM(R5)   ; SAVE
004674  104155                        MOV     (R5),R5         ; TRAVERSE TO NEXT UNIT
004675                                ASSUME  U.NEXT,0        ; ASSUME U.NEXT OFFSET IS ZERO
                                      .IF     NE,U.NEXT-0
                                      .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                      .ENDC
004675  106205  007702                CMP     #FIRSTU,R5      ; SEE IF TRAVERSED ENTIRE RING
004677  054642                        BNE     MKLOOP          ; IF NOT, BRANCH
004700  000000                        RETURN                  ; RETURN TO CALLING PROGRAM
        004701                  AREAI   =       .               ; INITILIZATION OVERLAY AREA
                                .SBTTL  GMPARM - GET MASTER PARAMETERS (PATTERN 16) AND SET UP OVERLAY ADDRESSES
004701                          GMPARM:
                                ;
                                ;       GET MASTER PARAMETERS
                                ;
004701  104302  007774                MOV     OVSTRT,R2       ; GET STARTING ADDRESS OF OVERLAY (LO)
004703  104303  007775                MOV     OVSTRT+1,R3     ; GET STARTING ADDRESS OF OVERLAY (HI)
004705  104020  004261                MOV     R2,OTABLE+2     ; MOVE STARTING ADDRESS INTO OVERLAY TABLE
004707  104207  000036                MOV     #NUMOVL-1,R0    ; R0 HAS NUMBER OF OVERLAYS
004711  104205  004257                MOV     #OTABLE,R5      ; R5 POINTS TO OVERLAY TABLE
004713  104651  000003          MORE: MOV     3(R5),R1        ; R1 HAS OVERLAY LENGTH
004715  110601                        ROR     R1              ; SHIFT TO MAKE BYTES
004716  103201  100001                BIC     #100001,R1      ; CLEAR UNUSED BITS
004720  105012                        ADD     R1,R2           ; FIND ADDRESS OF NEXT OVERLAY
004721  044723                        BCC     1$              ; IF NO CARRY, BRANCH
004722  115403                        INC     R3              ; PROPOGATE CARRY
004723  100652  000006          1$:   MOV     R2,6(R5)        ; STORE STARTING ADDRESS OF NEXT OVERLAY
004725  104651  000007                MOV     7(R5),R1        ; GET OVERLAY LENGTH (<<2)
004727  101031                        BIS     R3,R1           ; SET HI ORDER OVERLAY ADDRESS
004730  100651  000007                MOV     R1,7(R5)        ; SAVE
004732  105205  000004                ADD     #4,R5           ; POINT TO NEXT OVERLAY AREA
004734  117407                        DEC     R0              ; DECREMENT COUNT
004735  054713                        BNE     MORE            ; IF ALL OVERLAYS NOT SET UP, BRANCH
004736  104207  060003                MOV     #T4MPRM,R0      ; R0 CONTAINS HOST REQUEST
004740  021053                        CALL    HOSTRQ          ; INITIATE HOST REQUEST
004741  104207  001104                MOV     #HRQ.01,R0      ; R0 POINTS TO PATTERN INFORMATION
004743  104272                        MOV     (R0)+,R2        ; R2 HAS LENGTH OF PATTERN
```

```
004744  014754                          BEQ     3$              ; IF NO PATTERN, BRANCH
004745  104201  002257                  MOV     #PAT0+1,R1      ; R1 POINTS TO PATTERN 16 AREA (SKIP EDC)
004747  100212                          MOV     R2,(R1)+        ; MOVE PATTERN LENGTH TO PAT0 AREA
004750  104273                  2$:     MOV     (R0)+,R3        ; GET WORD OF PATTERN
004751  100213                          MOV     R3,(R1)+        ; MOVE TO PATTERN 16 AREA
004752  117402                          DEC     R2              ; DECREMENT COUNT
004753  054750                          BNE     2$              ; IF COUNT UNEXHAUSTED, BRANCH
004754  000000                  3$:     RETURN                  ; RETURN TO CALLING PROGRAM
                                .SBTTL  GETU   - POLL ALL PORTS, THEN GET UNITS TO TEST
004755                          GETU:
                                ;
                                ;       GET THE UNITS TO TEST
                                ;
                                ;
                                ;       POLL ALL PORTS AND FILL IN A UDA PORT INFORMATION TABLE (UNITS)
                                ;
004755  104205  000001                  MOV     #1,R5           ; MOVE INITIAL MASK TO R5
004757  104204  005345                  MOV     #UNITS,R4       ; R4 POINTS TO UNIT TABLE
004761                          5$:     PUSH    R4              ; SAVE R4
                                .IRP X,<R4>
                                                                                MOV X,-(SP)
                                .ENDR
                                                                                MOV R4,-(SP)
004761  100464                          MOV     R5,R2           ; MOVE MASK TO R2
004762  104052                          CALL    RDSTAT          ; GET DRIVE'S STATUS
004763  021007                          TST     R2              ; SEE IF ERROR
004764  115002                          BEQ     20$             ; IF NOT, BRANCH
004765  014777                          BMI     10$             ; IF MSB, PARITY ERRORS > HALF SECOND
004766  074772                          MOV     #SER10,R3       ; NO DRIVE ATTACHED
004767  104203  005205                  BR      15$             ; BRANCH
004771  004774                  10$:    MOV     #SER45,R3       ; PARITY ERRORS > HALF SECOND
004772  104203  002243          15$:    MOV     R3,1(R4)        ; SAVE ERROR MESSAGE
004774  100643  000001                  BR      100$            ; REPORT
004776  005171                  20$:    CLR     R3              ; SET UP TIMEOUT COUNT
004777  114003                  25$:    MOV     R5,R2           ; MOVE PORT SELECT TO R2
005000  104052                          CALL    RDSTAT          ; GET STATUS
005001  021007                          TST     R2              ; SEE IF ERROR
005002  115002                          BNE     30$             ; IF SO, BRANCH
005003  055007                          BIT     #RCVRDY,R1      ; SEE IF RECEIVER READY ASSERTED
005004  102201  000001                  BNE     35$             ; IF SO, BRANCH
005006  055016                  30$:    DEC     R3              ; DECREMENT COUNT
005007  117403                          BNE     25$             ; IF INCOMPLETE, BRANCH
005010  055000                          MOV     #SER11,R3       ; RECEIVER READY NEVER ASSERTED
005011  104203  005220                  MOV     R3,1(R4)        ; SAVE ERROR MESSAGE
005013  100643  000001                  BR      100$            ; REPORT
005015  005171                  35$:    BIT     #AVAIL,R1       ; SEE IF DRIVE IS AVAILABLE
005016  102201  000100                  BNE     40$             ; IF SO, BRANCH
005020  055026                          MOV     #SER40,R3       ; GET SECONDARY ERROR
005021  104203  005435                  MOV     R3,1(R4)        ; SAVE
005023  100643  000001                  BR      100$            ; EXIT
005025  005171                  40$:    MOV     #MAXSND,R2      ; SET UP MAXIMUM TRIES AT SENDING
005026  104202  001750                  MOV     #CR.GST,R3      ; R3 POINTS TO GET STATUS COMMAND
005030  104203  001617          45$:    MOV     (R3),R0         ; SET ADR OF SDI COMMAND BUFFER
005032  104137                          ASSUME  L2.OPC,0
005033                                  .IF     NE,L2.OPC-0
                                        .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                        .ENDC
```

GETU - POLL ALL PORTS, THEN GET UNITS TO TEST

```
005033  104631  000001                  MOV     L2.SLN(R3),R1   ; SET BUFFER LENGTH
005035                                  PUSH    R2              ; SAVE R2
                                        .IRP X,<R2>
                                                                                MOV X,-(SP)
                                        .ENDR
005035  100462                                                                  MOV R2,-(SP)
005036  104052                          MOV     R5,R2           ; SETUP FOR SEND
005037  060004                          XFC     SEND            ; SEND COMMAND
005040                                  POP     R2              ; RESTORE COUNT
                                        .IRP X,<R2>
                                                                                MOV (SP)+,X
                                        .ENDR
005040  104262                                                                  MOV (SP)+,R2
005041  115001                          TST     R1              ; DID UNIT ACCEPT COMMAND
005042  015052                          BEQ     50$             ; IF SO, BRANCH
005043  117402                          DEC     R2              ; DECREMENT COUNT
005044  055032                          BNE     45$             ; IF UNEXPIRED, BRANCH
005045  104203  005236                  MOV     #SER12,R3       ; GET ERROR NUMBER
005047  100643  000001                  MOV     R3,1(R4)        ; SAVE
005051  005171                          BR      100$
005052                          50$:    PUSH    R4              ; SAVE R4
                                        .IRP X,<R4>
                                                                                MOV X,-(SP)
                                        .ENDR
005052  100464                                                                  MOV R4,-(SP)
005053  104204  000003                  MOV     #3,R4           ; SET UP SHORT TIMEOUT
005055  104207  001702          55$:    MOV     #ST,R0          ; SET DATA BUFFER ADDRESS
005057  104631  000002                  MOV     L2.RLN(R3),R1   ; SET BUFFER LENGTH
005061  104052                          MOV     R5,R2           ; SETUP FOR RECEIVE
005062  060005                          XFC     RCV             ; RECEIVE SDI COMMAND
005063  115001                          TST     R1              ; DID ERROR OCCUR
005064  015124                          BEQ     85$             ; IF NOT, BRANCH
005065  106201  000001                  CMP     #1,R1           ; SEE IF TIMEOUT
005067  055076                          BNE     60$             ; IF NOT, BRANCH
005070  117404                          DEC     R4              ; DECREMENT TIMEOUT VALUE
005071  055055                          BNE     55$             ; IF NOT TIMEOUT, BRANCH
005072                                  POP     R4              ; RESTORE R4
                                        .IRP X,<R4>
                                                                                MOV (SP)+,X
                                        .ENDR
005072  104264                                                                  MOV (SP)+,R4
005073  104203  005250                  MOV     #SER13,R3       ; GET ERROR NUMBER
005075  005121                          BR      80$             ; BRANCH TO EXIT
005076                          60$:    POP     R4              ; RESTORE R4
                                        .IRP X,<R4>
                                                                                MOV (SP)+,X
                                        .ENDR
005076  104264                                                                  MOV (SP)+,R4
005077  110601                          ROR     R1              ; ROTATE INTO POSITION TO TEST
005100  110601                          ROR     R1              ; SEE IF FIRST WORD NOT START FRAME
005101  045105                          BCC     65$             ; IF NOT, BRANCH
005102  104203  005263                  MOV     #SER14,R3       ; GET ERROR NUMBER
005104  005121                          BR      80$             ; BRANCH TO END OF LOOP
005105  110601                  65$:    ROR     R1              ; SEE IF FRAMING ERROR
005106  045112                          BCC     70$             ; IF NOT, BRANCH
005107  104203  005311                  MOV     #SER15,R3       ; GET ERROR NUMBER
005111  005121                          BR      80$             ; BRANCH TO END OF LOOP
```

```
005112  110601                  70$:    ROR     R1              ; SEE IF CHECKSUM ERROR
005113  045117                          BCC     75$             ; IF NOT, BRANCH
005114  104203  005333                  MOV     #SER16,R3       ; GET ERROR NUMBER
005116  005121                          BR      80$             ; BRANCH TO END OF LOOP
005117  104203  005356          75$:    MOV     #SER17,R3       ; GET ERROR NUMBER
005121  100643  000001          80$:    MOV     R3,1(R4)        ; SAVE
005123  005171                          BR      100$            ; BRANCH TO END OF LOOP
                                ;
                                ;       NOW FILL IN THE TABLE WITH ALL THE SUBUNIT NUMBERS
                                ;
005124                          85$:    POP     R4              ; RESTORE R4
                                        .IRP X,<R4>
                                                                                        MOV (SP)+,X
                                        .ENDR
005124  104264                                                                          MOV (SP)+,R4
005125  104207  177777                  MOV     #-1,R0          ; GET 'NO UNITS' FLAG
005127  100647  000001                  MOV     R0,1(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
005131  100647  000002                  MOV     R0,2(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
005133  104307  001702                  MOV     ST,R0           ; R0 HAS UNIT NUMBER
005135  104072                          MOV     R0,R2           ; COPY R0 TO R2
005136  103207  170000                  BIC     #^CHBHINB,R0    ; R0 HAS UNIT NUMBER
005140                                  PUSH    <R1,R2>         ; SAVE R1 AND R2
                                        .IRP X,<R1,R2>
                                                                                        MOV X,-(SP)
                                        .ENDR
005140  100461                                                                          MOV R1,-(SP)
005141  100462                                                                          MOV R2,-(SP)
005142  104052                          MOV     R5,R2           ; MOVE UDA PORT MASK TO R2
005143  021007                          CALL    RDSTAT          ; GET STATUS
005144  102201  000002                  BIT     #ATTN,R1        ; SEE IF SPINABLE
005146  055151                          BNE     90$             ; IF SO, BRANCH
005147  101207  010000                  BIS     #10000,R0       ; SET 'NOT SPINABLE' FLAG
005151                          90$:    POP     <R2,R1>         ; RESTORE
                                        .IRP X,<R2,R1>
                                                                                        MOV (SP)+,X
                                        .ENDR
005151  104262                                                                          MOV (SP)+,R2
005152  104261                                                                          MOV (SP)+,R1
005153  101207  040000                  BIS     #40000,R0       ; FLAG UNIT AS NOT TESTED
005155  110702                          SWAB    R2              ; SWAP R2'S BYTES
005156  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
005157  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
005160  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
005161  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
005162  103202  177760                  BIC     #LBLONB,R2      ; CLEAR ALL BUT SUBUNIT BITS
005164  110602                  95$:    ROR     R2              ; MOVE SUBUNIT BIT TO CARRY
005165  045171                          BCC     100$            ; IF NO MORE SUBUNITS, BRANCH
005166  100247                          MOV     R0,(R4)+        ; MOVE SUBUNIT NUMBER TO TABLE
005167  115407                          INC     R0              ; INCREMENT SUBUNIT NUMBER
005170  005164                          BR      95$             ; BRANCH


005171                          100$:   POP     R4              ; R4 POINTS TO START OF UNIT JUST HANDLED
                                        .IRP X,<R4>
                                                                                        MOV (SP)+,X
                                        .ENDR
```

```
005171  104264                                                          MOV (SP)+,R4
005172  105204  000004          ADD     #4,R4           ; R4 WILL POINT TO NEXT UNIT (CLEAR CARRY FOR ROL)
005174  110205                  ROL     R5              ; R5 HAS NEXT UNIT PORT MASK
005175  106205  000020          CMP     #20,R5          ; SEE IF ALL PORTS TESTED
005177  054761                  BNE     5$              ; IF NOT, BRANCH
                        ;
                        ;       NOW GET THE PLUG NUMBERS TO TEST AND FIND THEM IN THE TABLE
                        ;
005200  104207  060012          MOV     #UTOTST,R0      ; GET WHAT SUBUNIT NUMBERS TO TEST REQUEST
005202  021053                  CALL    HOSTRQ          ; GET THE PLUG NUMBERS
005203  104207  001104          MOV     #HRQ.01,R0      ; R0 POINTS TO UNIT NUMBERS TO TEST
005205  104201  005345  105$:    MOV     #UNITS,R1       ; R1 POINTS TO UDA PORT INFORMATION
005207  104112          110$:   MOV     (R1),R2         ; R2 HAS UNIT
005210  075233                  BMI     130$            ; IF NONE, BRANCH
005211                          PUSH    R1              ; SAVE R1
                                .IRP X,<R1>
                                                                        MOV X,-(SP)
                                .ENDR
005211  100461                                                          MOV R1,-(SP)
005212  103202  160000  115$:    BIC     #160000,R2      ; CLEAR 'NOT TESTED', LEAVE 'UNSPINABLE' SET
005214  106172                  CMP     (R0),R2         ; SEE IF IT IS A UNIT TO TEST
005215  055221                  BNE     120$            ; NO MATCH
005216  100112                  MOV     R2,(R1)         ; SAVE UNIT AS ONE TO TEST
005217                          POP     R1              ; RESTORE STACK
                                .IRP X,<R1>
                                                                        MOV (SP)+,X
                                .ENDR
005217  104261                                                          MOV (SP)+,R1
005220  005335                  BR      175$            ; LOOK FOR THE NEXT ONE
005221  115401          120$:   INC     R1              ; POINT TO NEXT SUBUNIT
005222  104012                  MOV     R1,R2           ; COPY TO R2
005223  107202  005345          SUB     #UNITS,R2       ; SUBTRACT STARTING ADDRESS
005225  102202  000003          BIT     #3,R2           ; SEE IF STILL ON SAME UNIT
005227  015232                  BEQ     125$            ; IF NOT, BRANCH
005230  104112                  MOV     (R1),R2         ; SEE IF ANY MORE SUBUNITS
005231  035212                  BPL     115$            ; IF SO, BRANCH
005232                  125$:   POP     R1              ; RESTORE R1
                                .IRP X,<R1>
                                                                        MOV (SP)+,X
                                .ENDR
005232  104261                                                          MOV (SP)+,R1
005233  105201  000004  130$:    ADD     #4,R1           ; LOOK AT NEXT UNIT
005235  106201  005365          CMP     #UNITS+16.,R1   ; SEE IF ENTIRE TABLE SEARCHED
005237  055207                  BNE     110$            ; IF NOT, BRANCH
                        ;
                        ;       DIDN'T FIND THE REQUESTED UNITS -- DUMP ALL KNOWLEDGE OF THE
                        ;       UDA PORTS AND DIE
                        ;
005240  104170  001106          MOV     (R0),HRQ.03     ; SAVE UNIT NUMBER IN REQUEST BUFFER
005242  104204  001110          MOV     #HRQ.05,R4      ; R4 POINTS TO OUTPUT BUFFER
005244  104205  005345          MOV     #UNITS,R5       ; R5 POINTS TO UNIT TABLE
005246  104157          135$:   MOV     (R5),R0         ; GET FIRST WORD OF UNIT
005247  035254                  BPL     140$            ; IF VALID UNIT WAS FOUND, BRANCH
005250  104657  000001          MOV     1(R5),R0        ; GET POINTER TO ERROR MESSAGE
005252  100247                  MOV     R0,(R4)+        ; SAVE IN OUTPUT BUFFER
005253  005307                  BR      170$            ; EXIT
005254                  140$:   PUSH    R5              ; SAVE POINTER TO UNIT TABLE
```

```
                                        .IRP X,<R5>
                                                                                MOV X,-(SP)
                                        .ENDR
                                                                                MOV R5,-(SP)
005254  100465
005255  102207  010000                  BIT     #10000,R0       ; SEE IF DRIVE UNSPINABLE
005257  015263                          BEQ     145$            ; IF SPINABLE, BRANCH
005260  104207  005457                  MOV     #SER41,R0       ; REPORT DRIVE(S) UNSPINABLE
005262  005265                          BR      150$            ; BRANCH
005263  104207  005410          145$:   MOV     #SER18,R0       ; GET POINTER TO ERROR MESSAGE
005265  100247          150$:   MOV     R0,(R4)+        ; SAVE
005266  114007                  CLR     R0              ; CLEAR COUNT
005267  104251          155$:   MOV     (R5)+,R1        ; GET UNIT NUMBER
005270  075275                  BMI     160$            ; IF INVALID, BRANCH
005271  115407                  INC     R0              ; INCREMENT COUNT
005272  106207  000004          CMP     #4,R0           ; SEE IF MAX
005274  055267                  BNE     155$            ; IF NOT, LOOP
005275  104671  005340          160$:   MOV     SER18E-1(R0),R1 ; GET POINTER TO CORRECT ERROR MESSAGE
005277  100241                  MOV     R1,(R4)+        ; MOVE INTO OUTPUT BUFFER
005300                          POP     R5              ; RESTORE R5
                                        .IRP X,<R5>
                                                                                MOV (SP)+,X
                                        .ENDR
005300  104265                                                                  MOV (SP)+,R5
005301                          PUSH    R5              ; SAVE R5
                                        .IRP X,<R5>
                                                                                MOV X,-(SP)
                                        .ENDR
005301  100465                                                                  MOV R5,-(SP)
005302  104251          165$:   MOV     (R5)+,R1        ; GET SUBUNIT NUMBER
005303  100241                  MOV     R1,(R4)+        ; SAVE
005304  117407                  DEC     R0              ; DECREMENT COUNT
005305  055302                  BNE     165$            ; IF COUNT INCOMPLETE, BRANCH
005306                          POP     R5              ; RESTORE R5
                                        .IRP X,<R5>
                                                                                MOV (SP)+,X
                                        .ENDR
005306  104265                                                                  MOV (SP)+,R5
005307  105205  000004          170$:   ADD     #4,R5           ; POINT TO NEXT UNIT TABLE
005311  106205  005365                  CMP     #UNITS+16.,R5   ; SEE IF ENTIRE TABLE SEARCHED
005313  055246                          BNE     135$            ; IF NOT, BRANCH
005314                                  DEVFTL  1000            ; REPORT FATAL ERROR (WILL SHOW UP AS A 5000)
005314                                  ERROR   FTLDEV,1000,<>
        000012                          .RADIX  10
        000005                  NUMPTR  =       5
005314                          MOVMSG  #ER1000,4

                                        .IF     LT,4-10
005314  104200  005066  001107                                        MOV     #ER1000,HRQ.04
                                        .IFF
                                                                        MOV     #ER1000,HRQ.4
                                        .ENDC
                                        .IF     NB,<>
                                        .IRP    X,<>
                                MOVMSG  X,\NUMPTR
                                NUMPTR  =       NUMPTR + 1
                                        .ENDR
                                        .ENDC
```

```
005317  104202  051610                                          MOV     #1000!FTLDEV+4000.,R2
005321  104020  001105                                          MOV     R2,HRQ.02
005323  104200  005323  001104                                  MOV     #.,HRQ.01
        000010                  .RADIX
005326  104200  060014  001103                                  MOV     #ERRMC,HRQ.RQ
005331  104307  001103          MOV     HRQ.RQ,R0       ; SET UP FOR REPORT
005333  021053                  CALL    HOSTRQ          ; REPORT ERROR
005334  002573                  BR      STOP            ; END TEST

005335  115407          175$:   INC     R0              ; POINT TO NEXT UNIT TO TEST
005336  104172                  MOV     (R0),R2         ; CHECK NEXT UNIT
005337  035205                  BPL     105$            ; FIND IN UDA PORT INFORMATION
005340  000000                  RETURN                  ; RETURN TO INITIALIZATION CODE

005341  005432          SER18E: .WORD   SER18A          ; FOR MULTIPLE SUBUNIT ERROR REPORTING
005342  005426                  .WORD   SER18B
005343  005422                  .WORD   SER18C
005344  005416                  .WORD   SER18D

005345  000020          UNITS:  .REPT   16.
                                .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
                                .ENDR
005345  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005346  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005347  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005350  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005351  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005352  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005353  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005354  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005355  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005356  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005357  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005360  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005361  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005362  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005363  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
005364  177777                  .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
                        .SBTTL  BLDUNT - BUILD THE UNIT PARAMETER BLOCK
005365                  BLDUNT:
                        ;
                        ;       BLDUNT WILL BUILD THE UNIT AND SUBUNIT PARAMETERS
                        ;
005365  104202  000001          MOV     #1,R2           ; PORT 1 MASK
005367  104020  004500          MOV     R2,UMASK        ; SAVE MASK
005371  104204  005345          MOV     #UNITS,R4       ; POINT TO UNITS TO BE TESTED TABLE
005373  104147          ULOP:   MOV     (R4),R0         ; GET UNIT TO BE TESTED
005374  075456                  BMI     NOU             ; IF FIRST UNIT NOT TO BE TESTED, BRANCH
005375  102207  040000          BIT     #40000,R0       ; SEE IF FIRST SUBUNIT TO BE TESTED
005377  015417                  BEQ     3$              ; IF SO, BRANCH
005400  104647  000001          MOV     1(R4),R0        ; GET SECOND SUBUNIT INFORMATION
005402  102207  040000          BIT     #40000,R0       ; SEE IF THIS SUBUNIT IS USED
005404  015417                  BEQ     3$              ; IF SO, BRANCH
005405  104647  000002          MOV     2(R4),R0        ; SEE IF THIRD SUBUNIT TO BE TESTED
005407  102207  040000          BIT     #40000,R0       ; CHECK TESTING BIT
005411  015417                  BEQ     3$              ; IF IT IS TO BE TESTED, BRANCH
005412  104647  000003          MOV     3(R4),R0        ; SEE IF FOURTH SUBUNIT TO BE TESTED
```

```
005414  102207  040000                  BIT     #40000,R0       ; TEST TESTING BIT
005416  055456                          BNE     NOU             ; IF UNIT (THE WHOLE THING) NOT TO BE TESTED, BRANCH
005417  104207  000072          3$:     MOV     #TLEN.U,R0      ; R0 HAS UNIT DATA STRUCTURE LENGTH
005421  024616                          CALL    GETMEM          ; GET MEMORY
005422  100707  177053                  MOV     R0,@LASTU       ; LINK LAST POINTER TO THIS ONE
005424                                  ASSUME  U.NEXT,0
                                        .IF     NE,U.NEXT-0
                                        .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                        .ENDC
005424  104070  004477                  MOV     R0,LASTU        ; SAVE POINTER TO THIS ONE
005426  100672  000025                  MOV     R2,U.MASK(R0)   ; SAVE PORT SELECT MASK
005430  104203  000004                  MOV     #4,R3           ; MAXIMUM OF FOUR SUBUNITS
005432                                  PUSH    R0              ; SAVE POINTER TO UNIT INFORMATION
                                        .IRP X,<R0>
                                                                                        MOV X,-(SP)
                                        .ENDR
                                                                                        MOV R0,-(SP)
005432  100467                          MOV     (R4),R5         ; GET STARTING SUBUNIT NUMBER
005433  104145                          BIC     #^CHBHINB,R5    ; CLEAR 'NOT USED' BIT, IF SET
005434  103205  170000                  MOV     R5,U.UNUM(R0)   ; SAVE STARTING SUBUNIT NUMBER
005436  100675  000063                  ADD     #U.SUBP,R0      ; R0 WILL POINT TO SUBUNIT POINTERS
005440  105207  000001          1$:     MOV     (R4)+,R5        ; MOVE SUBUNIT NUMBER TO UNIT PARAMETERS
005442  104245                          BIT     #40000,R5       ; SEE IF SUBUNIT TO BE TESTED
005443  102205  040000                  BEQ     2$              ; IF SO, BRANCH
005445  015450                          MOV     #-1,R5          ; FLAG SUBUNIT AS NOT TESTED
005446  104205  177777          2$:     MOV     R5,(R0)+        ; MOVE TO SUBUNIT POINTERS (NON-NEG IF TESTED)
005450  100275                          DEC     R3              ; DECREMENT COUNT
005451  117403                          BNE     1$              ; IF ALL SUBUNITS ARE FILLED IN, BRANCH
005452  055442                          POP     R0              ; RESTORE R0
005453                                  .IRP X,<R0>
                                                                                        MOV (SP)+,X
                                        .ENDR
005453  104267                                                                          MOV (SP)+,R0
005454  107204  000004                  SUB     #4,R4           ; R4 NOW POINTS TO BEGINNING OF UNIT HANDLED
005456  104302  004500          NOU:    MOV     UMASK,R2        ; GET UNIT PORT MASK
005460  110202                          ROL     R2              ; ROTATE TO NEXT PORT
005461  104020  004500                  MOV     R2,UMASK        ; SAVE MASK
005463  105204  000004                  ADD     #4,R4           ; R4 POINTS TO NEXT UNIT INFORMATION
005465  106204  005365                  CMP     #UNITS+16.,R4   ; SEE IF ALL UNITS SET UP
005467  055373                          BNE     ULOP            ; IF ALL UNITS NOT SETUP, BRANCH
005470  104203  007702                  MOV     #FIRSTU,R3      ; COMPLETE RING
005472  100703  177003                  MOV     R3,@LASTU       ; LAST UNIT NOW POINTS TO FIRST
005474                                  ASSUME  U.NEXT,0
                                        .IF     NE,U.NEXT-0
                                        .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                        .ENDC
005474  104207  007702                  MOV     #FIRSTU,R0      ; R0 POINTS TO FIRST UNIT PARAMETERS
005476                          SUSETL: PUSH    R0              ; SAVE POINTER TO UNIT PARAMETERS
                                        .IRP X,<R0>
                                                                                        MOV X,-(SP)
                                        .ENDR
005476  100467                                                                          MOV R0,-(SP)
                                ;
                                .SBTTL  BLDSUS - BUILD ALL SUBUNIT PARAMETER BLOCKS ON THIS UNIT
                                ;BLDSUS
                                ;
                                ;       BUILD ALL SUBUNIT PARAMETERS
```

```
                                        ;
005477  104204  000001                SLOP: MOV     #U.SUBP,R4      ; R4 WILL POINT TO SUBUNIT POINTERS
005501  105074                              ADD     R0,R4           ; R4 POINTS TO SUBUNIT POINTERS
005502  104670  000063  004476              MOV     U.UNUM(R0),UCURSR       ; SAVE SUBUNIT NUMBER
005505  104205  000004                      MOV     #4,R5           ; MAXIMUM NUMBER OF SUBUNITS
005507  104147                              MOV     (R4),R0         ; R0 GETS SUBUNIT 'POINTER'
005510  076004                              BMI     NOSUN           ; IF SUBUNIT INACTIVE, BRANCH
005511  104070  004476                      MOV     R0,UCURSR       ; UCURSR GETS SUBUNIT NUMBER
005513                                      PUSH    <R3,R4,R5>      ; SAVE R3 - R5
                                        .IRP X,<R3,R4,R5>
                                                                                MOV X,-(SP)
                                        .ENDR
005513  100463                                                                 MOV R3,-(SP)
005514  100464                                                                 MOV R4,-(SP)
005515  100465                                                                 MOV R5,-(SP)
                                        .SBTTL  BLDBES - FIND HOW MANY WORDS NEEDED FOR THE BEGIN/END OR TRACK/GROUP SETS
                                        ;BLDBES
                                        ;
                                        ;       COMPUTE THE BEGIN/END SET AREA NEEDED
                                        ;
005516  104300  004476  001104          MOV     UCURSR,HRQ.01   ; MOVE SUBUNIT NUMBER TO COMMUNICATION AREA
005521  104207  060004                  MOV     #T4UPRM,R0      ; MOVE REQUEST NUMBER TO R0
005523  021053                          CALL    HOSTRQ          ; REQUEST INFORMATION FROM HOST
005524  102200  000040  001104          BIT     #BEUSED,HRQ.01  ; SEE IF BEGIN/END SETS ARE USED
005527  055535                          BNE     2$              ; IF SO, BRANCH
005530  104207  000020                  MOV     #S.TGSS+1,R0    ; MAXIMUM CONFIGURATION LENGTH
005532  105307  001112                  ADD     HRQ.07,R0       ; ADD NUMBER OF T/G SETS
005534  005546                          BR      BLDBB           ; BRANCH
005535  104207  000013          2$:     MOV     #S.BESS,R0      ; MINIMUM CONFIGURATION WORD COUNT
005537  104301  001106                  MOV     HRQ.03,R1       ; GET NUMBER OF BEGIN/END SETS
005541  055543                          BNE     1$              ; IF NON-ZERO, BRANCH
005542  115401                          INC     R1              ; MAKE R1 1
005543  105011          1$:             ADD     R1,R1           ; MAKE B/E SETS * 2
005544  105011                          ADD     R1,R1           ; MAKE B/E SETS * 4
005545  105017                          ADD     R1,R0           ; ADD TO LENGTH OF SUBUNIT PARAMETERS
                                        .SBTTL  BLDBB  - FIND HOW MANY WORDS NEEDED FOR THE BAD BLOCKS
005546                                  BLDBB:
                                        ;
                                        ;       COUNT THE BAD BLOCK PARAMETERS
                                        ;
005546                                  PUSH    R0              ; SAVE SUBUNIT LENGTH
                                        .IRP X,<R0>
                                                                                MOV X,-(SP)
                                        .ENDR
005546  100467                                                                 MOV R0,-(SP)
005547  104300  004476  001104          MOV     UCURSR,HRQ.01   ; MOVE SUBUNIT NUMBER TO COMMUNICATION AREA
005552  104207  060005                  MOV     #T4BB1,R0       ; MOVE REQUEST NUMBER TO R0
005554  021053                          CALL    HOSTRQ          ; REQUEST INFORMATION FROM HOST
005555                                  POP     R0              ; RESTORE SUBUNIT LENGTH
                                        .IRP X,<R0>
                                                                                MOV (SP)+,X
                                        .ENDR
005555  104267                                                                 MOV (SP)+,R0
005556  104301  001104                  MOV     HRQ.01,R1       ; GET NUMBER OF BAD BLOCKS
005560  104010  004501                  MOV     R1,NUMBB        ; SAVE NUMBER OF BAD BLOCKS
005562  105011                          ADD     R1,R1           ; MAKE BAD BLOCKS * 2
005563  105017                          ADD     R1,R0           ; ADD TO SUBUNIT DATA STRUCTURE LENGTH
```

```
005564  024616                          CALL    GETMEM          ; GET REQUESTED AMOUNT OF MEMORY
                                .SBTTL  COPYSU - COPY ALL SUBUNIT PARAMETERS TO SUBUNIT BLOCK
                                ;COPYSU
                                ;
                                ;       COPY SUBUNIT PARAMETRS TO SUBUNIT BLOCK
                                ;
005565                          PUSH    R0              ; SAVE R0
                                .IRP X,<R0>
                                                                        MOV X,-(SP)
                                .ENDR
005565  100467                                                          MOV R0,-(SP)
005566  104300  004476  001104        MOV     UCURSR,HRQ.01   ; MOVE SUBUNIT NUMBER TO COMMUNICATION AREA
005571  104207  060004                MOV     #T4UPRM,R0      ; MOVE REQUEST NUMBER TO R0
005573  021053                        CALL    HOSTRQ          ; REQUEST INFORMATION FROM HOST
005574                          POP     R0              ; RESTORE R0
                                .IRP X,<R0>
                                                                        MOV (SP)+,X
                                .ENDR
005574  104267                                                          MOV (SP)+,R0
                                ;
                                ;       THESE ARE THE ONLY BITS THAT THE HOST SHOULD SEND TO THE TEST
                                ;
                        077177  VALBIT  =       INITW!DCYLS!ECCCHK!RONLY!WONLY!RTRIES!SEQSEK!BEUSED!TRACKS!WCHECK!WCHKAL!DAT
                                ;
005575  103200  100600  001104        BIC     #^CVALBIT,HRQ.01        ; CLEAR ALL BUT VALID BITS (SUBUNIT PARAMETERS)
005600  104301  001105                MOV     HRQ.02,R1       ; GET PATTERN NUMBER
005602  100671  000004                MOV     R1,S.PAT(R0)    ; SAVE
005604  104305  001104                MOV     HRQ.01,R5       ; GET UNIT PARAMETERS
005606  102205  020000                BIT     #DCYLS,R5       ; SEE IF DIAGNOSTIC CYLINDERS ARE USED
005610  015616                        BEQ     4$              ; IF NOT, BRANCH
005611  102205  004000                BIT     #RONLY,R5       ; SEE IF READ ONLY
005613  055616                        BNE     4$              ; IF SO, BRANCH
005614  101205  040000                BIS     #INITW,R5       ; FLAG UNIS AS INITIALLY WRITTEN
005616  102205  000040        4$:     BIT     #BEUSED,R5      ; SEE IF BEGIN/END SETS USED
005620  055663                        BNE     3$              ; IF SO, BRANCH
005621  100175                        MOV     R5,(R0)         ; SAVE IN SUBUNIT AREA
005622                                ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
                                .IF     NE,S.PARM-0
                                .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                .ENDC
005622  104201  000013                MOV     #S.BESS,R1      ; R1 WILL POINT TO BEGIN/END SET AREA
005624  105071                        ADD     R0,R1           ; R1 POINTS TO TRACK/GROUP AREA
005625  104202  001106                MOV     #HRQ.03,R2      ; R2 POINTS TO START/END CYL
005627  104225                        MOV     (R2)+,R5        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005630  100615  000002                MOV     R5,2(R1)        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005632  104225                        MOV     (R2)+,R5        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005633  100615  000003                MOV     R5,3(R1)        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005635  104225                        MOV     (R2)+,R5        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005636  100115                        MOV     R5,(R1)         ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005637  104225                        MOV     (R2)+,R5        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005640  100615  000001                MOV     R5,1(R1)        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005642  105201  000004                ADD     #4,R1           ; R1 POINTS AFTER LIMITING CYLINDERS
005644  104224                        MOV     (R2)+,R4        ; GET TRACK/GROUP COUNT
005645  104225                5$:     MOV     (R2)+,R5        ; GET WORD
005646  100215                        MOV     R5,(R1)+        ; SAVE
005647  117404                        DEC     R4              ; DECREMENT COUNT
005650  055645                        BNE     5$              ; IF INCOMPLETE, BRANCH
```

```
005651  104415                          MOV     -(R1),R5        ; GET LAST T/G
005652  101205  100000                  BIS     #100000,R5      ; MARK AS END OF LIST
005654  100115                          MOV     R5,(R1)         ; SAVE
005655  104201  000020                  MOV     #S.TGSS+1,R1    ; R1 WILL POINT TO START OF T/G LIST
005657  105071                          ADD     R0,R1           ; R1 POINTS TO START OF T/G LIST
005660  105301  001112                  ADD     HRQ.07,R1       ; R1 NOW POINTS TO AFTER T/G'S (FOR BAD BLOCKS)
005662  005721                          BR      COPBB           ; BRANCH
005663  104201  000013          3$:     MOV     #S.BESS,R1      ; R1 WILL POINT TO BEGIN/END SETS
005665  105071                          ADD     R0,R1           ; R1 POINTS TO BEGIN/END SETS
005666  104202  001106                  MOV     #HRQ.03,R2      ; POINT TO BEGIN/END SET COUNT
005670  104224                          MOV     (R2)+,R4        ; GET COUNT, SEE IF NON-ZERO
005671  055675                          BNE     2$              ; IF SO, BRANCH
005672  115404                          INC     R4              ; SKIP COUNT
005673  101205  000200                  BIS     #ONLYCL,R5      ; FLAG AS BEGIN/END CYLINDER SUPPLIED
005675  100175          2$:     MOV     R5,(R0)         ; SAVE IN SUBUNIT AREA
005676                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
                                .IF     NE,S.PARM-0
                                .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                .ENDC
005676  104225                  1$:     MOV     (R2)+,R5        ; GET LO STARTING CYL
005677  100615  000002                  MOV     R5,2(R1)        ; SAVE
005701  104225                          MOV     (R2)+,R5        ; GET HI STARTING CYL
005702  100615  000003                  MOV     R5,3(R1)        ; SAVE
005704  104225                          MOV     (R2)+,R5        ; GET LO ENDING CYL
005705  100115                          MOV     R5,(R1)         ; SAVE
005706  104225                          MOV     (R2)+,R5        ; GET HI ORDER ENDING CYL
005707  100615  000001                  MOV     R5,1(R1)        ; SAVE
005711  105201  000004                  ADD     #4,R1           ; R1 POINTS AFTER BEGIN/END CYLINDERS
005713  117404                          DEC     R4              ; DECREMENT COUNT
005714  055676                          BNE     1$              ; LOOP AND COPY
005715  104415                          MOV     -(R1),R5        ; MOVE EOL TO BEGIN/END LIST
005716  101205  100000                  BIS     #100000,R5      ; SET HI BIT TO FLAG EOL
005720  100215                          MOV     R5,(R1)+        ; SAVE
005721                          COPBB:  PUSH    R0              ; SAVE R0
                                .IRP X,<R0>
                                                                                MOV X,-(SP)
                                .ENDR
005721  100467                                                                  MOV R0,-(SP)
005722  104300  004476  001104          MOV     UCURSR,HRQ.01   ; MOVE TO COMMUNICATION AREA
005725  104207  060005                  MOV     #T4BB1,R0       ; MOVE REQUEST NUMBER TO R0
005727  021053                          CALL    HOSTRQ          ; REQUEST INFORMATION FROM HOST
005730                                  POP     R0              ; RESTORE R0
                                .IRP X,<R0>
                                                                                MOV (SP)+,X
                                .ENDR
005730  104267                                                                  MOV (SP)+,R0
005731  104202  001105                  MOV     #HRQ.02,R2      ; R2 POINTS AT BAD BLOCKS
005733  104304  004501                  MOV     NUMBB,R4        ; R4 IS NUMBER OF BAD BLOCKS
005735  015773                          BEQ     NOBB            ; IF NO BAD BLOCKS, BRANCH
005736  100671  000012                  MOV     R1,S.BADP(R0)   ; MOVE POINTER TO BAD BLOCKS TO SUB PARAM
005740  114004                          CLR     R4              ; START LOOP AT ZERO
005741  106204  000016          BBLOP:  CMP     #14.,R4         ; SEE IF SECOND BLOCK NEEDED
005743  055756                          BNE     NOEXTR          ; IF NOT, BRANCH
005744                                  PUSH    R0              ; SAVE R0
                                .IRP X,<R0>
                                                                                MOV X,-(SP)
                                .ENDR
```

```
005744  100467                                                                  MOV R0,-(SP)
005745  104300  004476  001104          MOV     UCURSR,HRQ.01   ; SAVE IN COMMUNICATION AREA
005750  104207  060006                  MOV     #T4BB2,R0       ; LOAD REQUEST NUMBER
005752  021053                          CALL    HOSTRQ          ; REQUEST FURTHER BLOCKS FROM HOST
005753                                  POP     R0              ; RESTORE R0
                                        .IRP X,<R0>
                                                                                MOV (SP)+,X
                                        .ENDR
005753  104267                                                                  MOV (SP)+,R0
005754  104202  001104                  MOV     #HRQ.01,R2      ; POINT R2 TO BAD BLOCKS
005756  104225                  NOEXTR: MOV     (R2)+,R5        ; GET BAD BLOCK
005757  100215                          MOV     R5,(R1)+        ; SAVE BAD BLOCK
005760  104225                          MOV     (R2)+,R5        ; GET BAD BLOCK
005761  100215                          MOV     R5,(R1)+        ; SAVE BAD BLOCK
005762  115404                          INC     R4              ; INCREMENT R4
005763  106304  004501                  CMP     NUMBB,R4        ; SEE IF ALL BLOCKS COPIED
005765  055741                          BNE     BBLOP           ; IF NOT, BRANCH
005766  104415                          MOV     -(R1),R5        ; GET LAST BAD BLOCK
005767  101205  100000                  BIS     #100000,R5      ; FLAG AS EOL
005771  100215                          MOV     R5,(R1)+        ; SAVE EOL
005772  005775                          BR      CPYBEX          ; BRANCH
005773  100674  000012          NOBB:   MOV     R4,S.BADP(R0)   ; FLAG AS NO BAD BLOCKS
005775  114004                  CPYBEX: CLR     R4              ; CLEAR R4
005776  100674  000007                  MOV     R4,S.SCHR(R0)   ; FLAG CHARACTERISTICS AS NOT YET DEFINED
                                        ;
006000                                  POP     <R5,R4,R3>      ; RESTORE R3 - R5
                                        .IRP X,<R5,R4,R3>
                                                                                MOV (SP)+,X
                                        .ENDR
006000  104265                                                                  MOV (SP)+,R5
006001  104264                                                                  MOV (SP)+,R4
006002  104263                                                                  MOV (SP)+,R3
006003  006006                          BR      BSUSEX          ; BRANCH
006004  104207  177777          NOSUN:  MOV     #-1,R0          ; TO FLAG SUBUNIT AS INACTIVE
006006  100247                  BSUSEX: MOV     R0,(R4)+        ; MOVE POINTER TO SUB PARAM TO UNIT PARAMS
006007  115400  004476                  INC     UCURSR          ; MOVE TO NEXT SUBUNIT
006011  117405                          DEC     R5              ; DECREMENT COUNT
006012  055507                          BNE     SLOP            ; IF UNEXPIRED, BRANCH
006013                                  POP     R0              ; RESTORE UNIT POINTER
                                        .IRP X,<R0>
                                                                                MOV (SP)+,X
                                        .ENDR
006013  104267                                                                  MOV (SP)+,R0
006014  104177                          MOV     (R0),R0         ; GO TO NEXT UNIT
006015                                  ASSUME  U.NEXT,0
                                        .IF     NE,U.NEXT-0
                                        .ERROR  ;       THE ASSUMPTION THAT THE ABOVE VALUES ARE = IS FALSE
                                        .ENDC
006015  103207  170000                  BIC     #UNADDR,R0      ; CLEAR UNUSED ADDRESSING BITS
006017  106207  007702                  CMP     #FIRSTU,R0      ; SEE IF THE ENTIRE RING IS SET UP
006021  055476                          BNE     SUSETL          ; IF NOT, BRANCH
006022  000000                          RETURN
                                        .IF     LE,MAXADR-.
                                MAXADR  =       .+1
                                        .ENDC
68                                      .IF     LE,MAXADR-.
69                              MAXADR  =       .+1
```

```
     70                                         .ENDC
     84 006023                                  TEXT
                                                .LIST   ME
                                        .SBTTL  ********** NON-LOADED OVERLAY, ERROR MESSAGES
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;
                                        ;       ERROR MESSAGE FORMAT BUFFERS (NEVER LOADED INTO UDA)
                                        ;
        006023                                  DMOVLY  MS,0                    ; ERROR MESSAGE OVERLAY
        006023                                  OVTERM          ;TERMINATE LAST OVERLAY
        006023  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                005110                  OVL.MN  =       .-OVE.MN
                005110                  OVL...  =       OVL...+OVL.MN
                005530                  OV...   =       OV...+OVL.MN
                                                .CLEDC          ;CLEAR EDC FOR NEXT OVERLAY
                                        .MACRO  OVTERM
                                                .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                        OVL.MS  =       .-OVE.MS
                                        OVL...  =       OVL...+OVL.MS
                                        OV...   =       OV...+OVL.MS
                                        .ENDM
                013260                  OVS.MS  =       OV...*2
                000000                  OVE.MS  =       0
                                                .OFSET  OV...-0
                                                .NLIST  BEX
                                                .LIST   MEB
        000000  042     101     124     ER1:    .ASCII  \"ATTN ASSERTED DURING SEEK'N\
        000016  042     123     105             .ASCII  \"SEEK FROM GRP 'D8'' CYL 'D28'' TO GRP 'D8'' CYL 'D28NR1\
        000050  000                             .BYTE   0
        000051  042     122     105     SER22:  .ASCII  \"REAL TIME STATE 'H16N\
        000064  042     123     124             .ASCII  \"STATUS (R TO L): 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
        000116  000                             .BYTE   0
        000117  042     101     124     ER2:    .ASCII  \"ATTN ASSERTED UNEXPECTEDLY, ASYNC DRIVE ERROR OR LOGGABLE'N\
        000155  042     111     116             .ASCII  \"INFORMATION'NR1\
        000165  000                             .BYTE   0
        000166  042     123     105     ER3:    .ASCII  \"SEEK DID NOT COMPLETE, NEITHER ATTN OR R/W RDY WAS ASSERTED'N\
        000225  042     102     105             .ASCII  \"BEFORE TIMEOUT'N\
        000235  042     123     105             .ASCII  \"SEEK FROM GRP 'D8'' CYL 'D28'' TO GRP 'D8'' CYL 'D28NR1\
        000270  000                             .BYTE   0
        000271  042     122     103     ER4:    .ASCII  \"RCT AREA CORRUPTED, COULD NOT FIND REPLACEMENT FOR'N\
        000323  042     114     102             .ASCII  \"LBN 'R1\
        000327  042     101     124             .ASCII  \"ATTEMPTED TO READ RCT LBN 'D28N\
        000347  042     123     105             .ASCII  \"SEARCING FOR LBN'S10D28N\
        000364  000                             .BYTE   0
        000365  042     110     105     ER5:    .ASCII  \"HEADER NOT FOUND DURING WRITE'N\
        000405  122     061     042             .ASCII  \R1''BN ''D28NR3\
        000413  042     123     105             .ASCII  \"SECTORS FROM INDEX 'D8'' TRK 'D8'' GRP 'D8'' CYL 'D28N\
        000445  042     117     122             .ASCII  \"ORIGIN OF SEEK: GRP 'D8'' CYL 'D28N\
        000467  000                             .BYTE   0
        000470  042     101     124     SER19:  .ASCII  \"ATTEMPT 'D3N\
        000476  122     061     042             .ASCII  \R1''BN ''D28NR3\
        000505  042     123     105             .ASCII  \"SECTORS FROM INDEX 'D8'' TRK 'D8'' GRP 'D8'' CYL 'D28N\
        000537  042     117     122             .ASCII  \"ORIGIN OF SEEK: GRP 'D8'' CYL 'D28NR1\
        000561  000                             .BYTE   0
```

```
000562  042  123  105  ER6:     .ASCII  \"SELECT TRACK AND WRITE LEVEL 1 CMD NOT EXECUTED"NR1\
000614  000                    .BYTE   0
000615  042  105  103  ER7:     .ASCII  \"ECC DETECTED ERROR"NR1\
000630  000                    .BYTE   0
000631  042  105  103  ER8:     .ASCII  \"ECC DETECTED ERROR, BUT CORRECTION FAILED"NR1\
000660  000                    .BYTE   0
000661  042  122  105  SER21:   .ASCII  \"RETRY "D4N\
000666  042  105  122           .ASCII  \"ERROR RECOVERY LEVEL "D8N\
000703  122  061  042           .ASCII  \R1"BN "D28NR3\
000712  042  123  105           .ASCII  \"SECTORS FROM INDEX "D8" TRK "D8" GRP "D8" CYL "D28N\
000744  000                    .BYTE   0
000745  042  105  103  ER9:     .ASCII  \"ECC CORRECTIONS EXCEED THRESHOLD"NR1\
000767  000                    .BYTE   0
000770  042  105  103  ER10:    .ASCII  \"ECC CORRECTION SUCCEEDED, BUT EDC DETECTS ERROR"NR1\
001022  042  105  104           .ASCII  \"EDC COMPUTED "O16N\
001033  042  105  104           .ASCII  \"EDC READ"S5O16N\
001043  000                    .BYTE   0
001044  042  122  105  ER11:    .ASCII  \"READ DID NOT SUCCEED ON ANY RECOVERY LEVEL"N\
001072  122  061  042           .ASCII  \R1"BN "D28NR3\
001101  042  107  122           .ASCII  \"GRP "D8" CYL "D28N\
001112  000                    .BYTE   0
001113  042  104  101  ER12:    .ASCII  \"DATA COMPARISON FAILED"NR1\
001130  122  061  042           .ASCII  \R1"BN "D28NR3\
001137  042  123  105           .ASCII  \"SECTORS FROM INDEX "D8" TRK "D8" GRP "D8" CYL "D28N\
001171  042  120  101           .ASCII  \"PATTERN NUMBER  "D4N\
001203  042  117  106           .ASCII  \"OFFSET OF ERROR WITHIN BUFFER: "D8N\
001225  042  117  106           .ASCII  \"OFFSET OF ERROR WITHIN DISPLAYED LIST: "D8" (1ST WORD OFFSET 0)"N\
001266  123  064  117           .ASCII  \S4016S4016S4016S4016S4016S4016N\
001306  123  064  117           .ASCII  \S4016S4016S4016S4016S4016S4016N\
001325  000                    .BYTE   0
001326  042  105  103  SER24:   .ASCII  \"ECC OR EDC HAD DETECTED ERROR IN BUFFER"N\
001353  000                    .BYTE   0
001354  042  105  103  SER25:   .ASCII  \"ECC OR EDC HAD <<NOT>> DETECTED ERROR IN BUFFER"N\
001405  000                    .BYTE   0
001406  042  104  122  ER13:    .ASCII  \"DRIVE NOT ONLINE TO UDA, AND NOT SPINABLE"N\
001434  000                    .BYTE   0
001435  042  125  116  ER14:    .ASCII  \"UNABLE TO COMPLETE SEEK -- TRIED 3 TIMES"N\
001462  122  061  042           .ASCII  \R1"BN "D28NR3\
001471  042  107  122           .ASCII  \"GRP "D8" CYL "D28N\
001502  000                    .BYTE   0
001503  042  123  105  ER15:    .ASCII  \"SEEK REQUIRED "D2" RETRIES BEFORE COMPLETING"N\
001532  122  061  042           .ASCII  \R1"BN "D28NR3\
001541  042  107  122           .ASCII  \"GRP "D8" CYL "D28N\
001552  000                    .BYTE   0
001553  042  105  122  ER16:    .ASCII  \"ERRORS DURING DRIVE INITIALIZATION AND SETUP"NR1\
001603  000                    .BYTE   0
001604  042  116  117  ER17:    .ASCII  \"NO VALID STATE FROM DRIVE"N\
001622  042  116  117           .ASCII  \"NO DRIVE CLOCKS"N\
001633  000                    .BYTE   0
001634  042  116  117  ER17A:   .ASCII  \"NO VALID STATE FROM DRIVE"N\
001652  042  110  101           .ASCII  \"HARD PARITY OR PULSE ERROR FOR 1/2 A SECOND"N\
001701  000                    .BYTE   0
001702  042  101  124  ER18:    .ASCII  \"ATTEMPT TO WRITE ON WRITE PROTECTED DRIVE"N\
001730  042  105  122           .ASCII  \"ERROR CODE RETURNED FROM UDA: "O16NR1\
001753  000                    .BYTE   0
001754  042  110  105  ER19:    .ASCII  \"HEADER NOT FOUND DURING READ"N\
001773  122  061  042           .ASCII  \R1"BN "D28NR3\
```

```
002002  042  123  105        .ASCII  \"SECTORS FROM INDEX '"D8" TRK '"D8" GRP '"D8" CYL '"D28N\
002034  042  117  122        .ASCII  \"ORIGIN OF SEEK: GRP '"D8" CYL '"D28N\
002055  000                  .BYTE   0
002056  042  123  105  ER20: .ASCII  \"SELECT TRACK AND READ LEVEL 1 CMD NOT SENT'"NR1\
002105  000                  .BYTE   0
002106  042  104  122  ER21: .ASCII  \"DRIVE NOT FORMATTED IN 512 BYTE MODE -- UNABLE TO TEST'"N\
002142  042  130  102        .ASCII  \"XBN 0 MODE WORD: '"O16N\
002156  000                  .BYTE   0
002157  042  125  116  ER23: .ASCII  \"UNABLE TO CONTINUE TESTING'"NR1R1\
002177  000                  .BYTE   0
002200  042  122  125  SER42: .ASCII  \"RUN/STOP SWITCH OUT'"N\
002213  000                  .BYTE   0
002214  042  123  120  SER43: .ASCII  \"SPINDLE DROPPED READY'"N\
002230  000                  .BYTE   0
002231  042  120  117  SER44: .ASCII  \"PORT SWITCH OUT'"N\
002242  000                  .BYTE   0
002243  042  120  101  SER45: .ASCII  \"PARITY ERRORS FOR MORE THAN A HALF SECOND'"N\
002271  000                  .BYTE   0
002272  042  105  104  ER24: .ASCII  \"EDC DETECTED ERROR BUT ECC DID NOT'"NR1\
002315  042  105  104        .ASCII  \"EDC COMPUTED '"O16" EDC READ '"O16N\
002336  000                  .BYTE   0
002337  042  127  122  ER25: .ASCII  \"WRITE ATTEMPTED MAXIMUM TIMES'"N\
002357  122  061  042        .ASCII  \R1'"BN '"D28NR3\
002365  000                  .BYTE   0
002366  042  122  105  ER26: .ASCII  \"READ ATTEMPTED MAXIMUM TIMES'"N\
002405  122  061  042        .ASCII  \R1'"BN '"D28NR3\
002414  000                  .BYTE   0
002415  042  102  117  ER28: .ASCII  \"BOTH READ ONLY <AND> WRITE ONLY BITS SET -- HOST ERROR'"N\
002451  000                  .BYTE   0
002452  042  125  116  ER33: .ASCII  \"UNABLE TO CORRECTLY READ OVERLAY '"O16NR1\
002476  000                  .BYTE   0
002477  042  124  110  SER39: .ASCII  \"THIS UDA AND ALL DRIVES ATTACHED WILL BE REMOVED FROM TESTING'"N\
002537  000                  .BYTE   0
002540  042  123  105  ER34: .ASCII  \"SERDES OVERRUN ERROR DURING READ'"NR1\
002562  000                  .BYTE   0
002563  042  104  101  ER35: .ASCII  \"DATA OR STATE CLOCK TIMEOUT DURING READ'"NR1\
002611  000                  .BYTE   0
002612  042  104  101  ER36: .ASCII  \"DATA SYNC TIMEOUT DURING READ'"NR1\
002633  000                  .BYTE   0
002634  042  122  057  ER37: .ASCII  \"R/W RDY DROPPED BEFORE/DURING READ'"NR1\
002657  000                  .BYTE   0
002660  042  122  103  ER38: .ASCII  \"RCVR RDY DROPPED BEFORE/DURING READ'"NR1\
002704  000                  .BYTE   0
002705  042  101  114  ER40: .ASCII  \"ALL COPIES OF RCT READ WITH ERROR, SEARCHING FOR'"N\
002736  042  114  102        .ASCII  \"LBN '"R1\
002742  042  114  101        .ASCII  \"LAST RCT LBN SEARCHED '"D28N\
002760  042  123  105        .ASCII  \"SEARCING FOR LBN'"S6D28N\
002774  000                  .BYTE   0
002775  042  103  117  ER41: .ASCII  \"COULD NOT FIND REPLACEMENT FOR'"N\
003015  042  114  102        .ASCII  \"LBN '"R1\
003021  042  114  102        .ASCII  \"LBN TO REPLACE '"D28N\
003034  000                  .BYTE   0
003035  042  124  110  SER26: .ASCII  \"THAT WAS REVECTORED'"N\
003050  000                  .BYTE   0
003051  042  127  111  SER32: .ASCII  \"WITH HEADER NOT FOUND'"N\
003065  000                  .BYTE   0
003066  042  124  111  ER42: .ASCII  \"TIMEOUT WAITING FOR SECTOR OR INDEX PULSE'"N\
```

```
003114  042  107  122          .ASCII  \"GRP '"D8" CYL '"D28NR1\
003126  000                    .BYTE   0
003127  042  123  105  ER44:   .ASCII  \"SEEK OR HEAD SELECT ERROR DETECTED DURING WRITE'"NR1\
003161  000                    .BYTE   0
003162  042  123  105  ER45:   .ASCII  \"SEEK OR HEAD SELECT ERROR DETECTED DURING READ'"NR1\
003213  000                    .BYTE   0
003214  042  104  101  ER47:   .ASCII  \"DATA OR STATE CLOCK TIMEOUT DURING WRITE'"NR1\
003242  000                    .BYTE   0
003243  042  122  057  ER48:   .ASCII  \"R/W RDY DROPPED BEFORE/DURING WRITE'"NR1\
003267  000                    .BYTE   0
003270  042  122  103  ER49:   .ASCII  \"RCVR RDY DROPPED BEFORE/DURING WRITE'"NR1\
003314  000                    .BYTE   0
003315  122  061  042  ER50:   .ASCII  \R1"'BEGIN/END SET STARTING BLOCK NUMBER GREATER THAN ENDING BLOCK NUMBER'"N\
003361  000                    .BYTE   0
003362  122  061  042  ER51:   .ASCII  \R1"'THE BEGIN/END SETS GIVEN OVERLAP'"N\
003404  000                    .BYTE   0
003405  122  061  042  ER52:   .ASCII  \R1"'BEGIN/END SET ENDING BLOCK NUMBER EXCEEDS MAXIMUM'"N\
003440  042  115  101          .ASCII  \"'MAXIMUM BLOCK NUMBER ON DEVICE IS '"D28N\
003464  000                    .BYTE   0
003465  122  061  042  ER53:   .ASCII  \R1"'DUPLICATE BAD BLOCKS'"N\
003501  000                    .BYTE   0
003502  122  061  042  ER54:   .ASCII  \R1"'BAD BLOCK NUMBER EXCEEDS MAXIMUM. MAXIMUM BLOCK NUMBER'"N\
003537  042  117  116          .ASCII  \"'ON DEVICE IS '"D28N\
003551  000                    .BYTE   0
003552  122  061  042  ER55:   .ASCII  \R1"'STARTING CYLINDER GREATER THAN ENDING CYLINDER'"N\
003603  000                    .BYTE   0
003604  122  061  042  ER56:   .ASCII  \R1"'RANDOM AND SEQUENTIAL SEEKS CAN NOT BE MIXED WITHIN A UNIT'"N\
003643  000                    .BYTE   0
003644  122  061  042  ER57:   .ASCII  \R1"'OVERFLOW WHEN CALCULATING THE L/DBN FROM THE GIVEN CYLINDER'"N\
003704  042  103  131          .ASCII  \"'CYLINDER TOO LARGE'"N\
003716  000                    .BYTE   0
003717  122  061  122  ER58:   .ASCII  \R1R1"' EXCEEDS MAXIMUM FOR DEVICE. MAXIMUM IS '"D8N\
003747  000                    .BYTE   0
003750  122  061  042  ER59:   .ASCII  \R1"'TWO IDENTICAL '"R1"'S'"N\
003764  000                    .BYTE   0
003765  122  061  122  ER62:   .ASCII  \R1R1"'BN COMPUTED FROM END CYLINDER GIVEN EXCEEDS MAXIMUM '"R1"'BN ON'"N\
004027  042  104  105          .ASCII  \"'DEVICE - CYLINDER TOO LARGE'"N\
004046  000                    .BYTE   0
004047  042  117  120  SER20:  .ASCII  \"'OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT'"N\
004114  000                    .BYTE   0
004115  042  122  105  ER63:   .ASCII  \"'REAL TIME STATE RECEIVE ERROR DURING WRITE'"NR1\
004144  000                    .BYTE   0
004145  042  122  105  ER64:   .ASCII  \"'REAL TIME STATE RECEIVE ERROR DURING READ'"NR1\
004174  000                    .BYTE   0
004175  042  125  116  ER68:   .ASCII  \"'UNKNOWN ERROR CODE DURING WRITE'"NR1\
004217  000                    .BYTE   0
004220  042  125  116  ER69:   .ASCII  \"'UNKNOWN ERROR CODE DURING READ'"NR1\
004241  000                    .BYTE   0
004242  042  105  122  SER36:  .ASCII  \"'ERROR CODE RETURNED '"O16NR1\
004260  000                    .BYTE   0
004261  042  124  111  ER70:   .ASCII  \"'TIMEOUT OF SEND'"NR1R1\
004274  000                    .BYTE   0
004275  042  124  111  ER71:   .ASCII  \"'TIMEOUT OF RECEIVE'"NR1R1\
004311  000                    .BYTE   0
004312  042  106  111  ER72:   .ASCII  \"'FIRST WORD RECEIVED WAS NOT START FRAME'"NR1R1\
004341  000                    .BYTE   0
004342  042  106  122  ER73:   .ASCII  \"'FRAMING ERROR ON LEVEL 0 RECEIVE'"NR1R1\
```

```
004365  000                             .BYTE   0
004366  042  103  110  ER74:   .ASCII  \"CHECKSUM ERROR ON LEVEL 0 RECEIVE"NR1R1\
004412  000                             .BYTE   0
004413  042  102  125  ER75:   .ASCII  \"BUFFER SIZE SMALLER THAN LEVEL 2 RESPONSE"NR1R1\
004443  000                             .BYTE   0
004444  042  122  105  ER76:   .ASCII  \"RESPONSE OF LEVEL 2 CMD NOT AS EXPECTED"NR1\
004472  042  105  130                  .ASCII  \"EXPECTED RESPONSE "H8N\
004505  042  122  105                  .ASCII  \"RESPONSE RECEIVED "H8NR1\
004522  000                             .BYTE   0
004523  042  104  122  ER77:   .ASCII  \"DRIVE NEVER DEASSERTED RECEIVER READY AFTER SEND"NR1R1\
004556  000                             .BYTE   0
004557  042  125  116  ER78:   .ASCII  \"UNKNOWN ERROR CODE RETURNED FROM LEVEL 2 RECEIVE"NR1\
004611  042  105  122                  .ASCII  \"ERROR CODE RETURNED "O16NR1\
004627  000                             .BYTE   0
004630  042  101  124  SER0:   .ASCII  \"ATTEMPTING TO GET STATUS"N\
004645  000                             .BYTE   0
004646  042  101  124  SER1:   .ASCII  \"ATTEMPTING DRIVE CLEAR CMD"N\
004664  000                             .BYTE   0
004665  042  101  124  SER2:   .ASCII  \"ATTEMPTING TO BRING DRIVE ONLINE"N\
004706  000                             .BYTE   0
004707  042  101  124  SER3:   .ASCII  \"ATTEMPTING TO CHANGE MODE"N\
004725  000                             .BYTE   0
004726  042  101  124  SER4:   .ASCII  \"ATTEMPTING ERROR RECOVERY CMD"N\
004746  000                             .BYTE   0
004747  042  101  124  SER5:   .ASCII  \"ATTEMPTING TO GET SUBUNIT CHAR"N\
004767  000                             .BYTE   0
004770  042  101  124  SER6:   .ASCII  \"ATTEMPTING TO SPIN UP DRIVE"N\
005007  000                             .BYTE   0
005010  042  101  124  SER7:   .ASCII  \"ATTEMPTING TO RECALIBRATE"N\
005026  000                             .BYTE   0
005027  042  101  124  SER8:   .ASCII  \"ATTEMPTING TO GET COMMON CHAR"N\
005047  000                             .BYTE   0
005050  042  101  124  SER9:   .ASCII  \"ATTEMPTING TO ISSUE SEEK"N\
005065  000                             .BYTE   0
005066  042  125  116  ER1000: .ASCII  \"UNABLE TO FIND REQUESTED DRIVE FOR TESTING"N\
005114  042  124  110                  .ASCII  \"THE FOLLOWING IS VISIBLE ON THE PORTS"N\
005140  042  125  104                  .ASCII  \"UDA PORT 0 -- "R1\
005151  042  125  104                  .ASCII  \"UDA PORT 1 -- "R1\
005162  042  125  104                  .ASCII  \"UDA PORT 2 -- "R1\
005173  042  125  104                  .ASCII  \"UDA PORT 3 -- "R1\
005204  000                             .BYTE   0
005205  042  116  117  SER10:  .ASCII  \"NO DRIVE ATTACHED"N\
005217  000                             .BYTE   0
005220  042  122  103  SER11:  .ASCII  \"RCVR RDY NEVER ASSERTED"N\
005235  000                             .BYTE   0
005236  042  124  111  SER12:  .ASCII  \"TIMEOUT OF SEND"N\
005247  000                             .BYTE   0
005250  042  124  111  SER13:  .ASCII  \"TIMEOUT OF RECEIVE"N\
005262  000                             .BYTE   0
005263  042  106  111  SER14:  .ASCII  \"FIRST WORD RECEIVED WAS NOT START FRAME"N\
005310  000                             .BYTE   0
005311  042  106  122  SER15:  .ASCII  \"FRAMING ERROR ON LEVEL 0 RECEIVE"N\
005332  000                             .BYTE   0
005333  042  103  110  SER16:  .ASCII  \"CHECKSUM ERROR ON LEVEL 0 RECEIVE"N\
005355  000                             .BYTE   0
005356  042  122  105  SER17:  .ASCII  \"RESPONSE LONGER THAN EXPECTED FOR GET STATUS CMD"N\
005407  000                             .BYTE   0
```

```
005410  042  104  122  SER18:  .ASCII  \"DRIVE "R1\
005415  000                   .BYTE   0
005416  104  061  062  SER18D: .ASCII  \D12",  "\      ; NOTE: THE STRING WITHIN THE \\'S <<MUST>>
005422  104  061  062  SER18C: .ASCII  \D12",  "\      ; BE AN EVEN NUMBER OF CHAR, BECAUSE THE
005426  104  061  062  SER18B: .ASCII  \D12",  "\      ; LABELS WILL FORCE WORD ALINGMENT, LEAVING
005432  104  061  062  SER18A: .ASCII  \D12N\          ; JUNK IN THE LAST BYTE. (SER18A OK TO BE ODD)
005434  000                   .BYTE   0
005435  042  104  122  SER40:  .ASCII  \"DRIVE NOT AVAILABLE TO THIS UDA"N\
005456  000                   .BYTE   0
005457  042  125  116  SER41:  .ASCII  \"UNSPINABLE DRIVE "R1\
005471  000                   .BYTE   0
005472  042  122  105  RBNTXT: .ASCII  \"REVECTORED TO RBN "D28N\
005506  000                   .BYTE   0
005507  042  124  122  TRK$:   .ASCII  \"TRACK"\
005512  000                   .BYTE   0
005513  042  107  122  GRP$:   .ASCII  \"GROUP"\
005516  000                   .BYTE   0
005517  042  114  042  L$:     .ASCII  \"L"\
005520  000                   .BYTE   0
005521  042  104  042  D$:     .ASCII  \"D"\
005522  000                   .BYTE   0
005523  042  122  103  RCTL$:  .ASCII  \"RCT L"\
005526  000                   .BYTE   0
                      .SBTTL  INFORMATIONAL MESSGES
                      ;
                      ;       MESSAGES
                      ;
005527  116  042  111  MS1:    .ASCII  \N"INITIAL WRITE COMPLETE"N\
005544  000                   .BYTE   0
005545  116  042  124  MS2:    .ASCII  \N"THE PREVIOUS DEVICE FATAL WILL CAUSE THE FOLLOWING DRIVES"N\
005603  042  124  117         .ASCII  \"TO BE DROPPED: "R1\
005615  000                   .BYTE   0
005616  116  042  101  MS3:    .ASCII  \N"A CORRECTABLE ECC ERROR EXISTS IN "R1"BN "D28N\
005646  042  123  105         .ASCII  \"SECTORS FROM INDEX "D8" TRK "D8" GRP "D8" CYL "D28N\
005700  000                   .BYTE   0
005701  116  042  122  MS4:    .ASCII  \N"READ ONLY DRIVE, INITIAL WRITE WILL NOT BE PERFORMED"N\
005735  000                   .BYTE   0
```

```
                                .SBTTL  ********** OVERLAY MODULE SETUP - OPERATION TO BE DONE THIS PASS
005735                                  DMOVLY  SU,AREA0
005736  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
000001                          SETUP   =       1               ; SETUP OVERLAY
                                ;
                                ;       SETUP CLEARS ALL ERROR BITS, THE ERROR COUNT, FINDS THE NEXT
                                ;       LBN TO BE READ OR WRITTEN, DESIDES TO READ OR WRITE, CALCULATES
                                ;       THE CYLINDER, TRACK AND GROUP THE LBN RESIDES ON; THEN CALLS
                                ;       SEEK.
                                ;
                                        .ENABL  LSB
004413  114002                          CLR     R2              ; NO ERRORS
004414  104657  000024                  MOV     U.CSEC(R5),R0   ; GET TOTAL NUMBER OF SECTORS ALLREADY R/W
004416  105657  000021                  ADD     U.NSEC(R5),R0   ; ADD HOW MANY R/W THIS PASS
004420  106657  000023                  CMP     U.TSEC(R5),R0   ; SEE IF ALL DONE
004422  014442                          BEQ     2$              ; IF SO, BRANCH (NEW OPERATION)
004423  100657  000024                  MOV     R0,U.CSEC(R5)   ; SAVE CURRENT COUNT
004425  104657  000053                  MOV     U.CBN(R5),R0    ; GET LBN
004427  105657  000021                  ADD     U.NSEC(R5),R0   ; ADD NUMBER OF SECTORS R/W THIS PASS
004431  100657  000053                  MOV     R0,U.CBN(R5)    ; SAVE
004433  044514                          BCC     9$              ; IF NO CARRY, BRANCH
004434  104657  000054                  MOV     U.CBN+1(R5),R0  ; GET HI LBN
004436  115407                          INC     R0              ; PROPOGATE CARRY
004437  100657  000054                  MOV     R0,U.CBN+1(R5)  ; SAVE
004441  004514                          BR      9$              ; BRANCH
004442  104657  000046          2$:     MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
004444  102207  000010                  BIT     #WCHECK,R0      ; SEE IF WRITE CHECK IS TO BE DONE
004446  054475                          BNE     5$              ; IF SO, BRANCH
004447  104141                          MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
004450                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
004450  102201  000100                  BIT     #SEQSEK,R1      ; SEE IF SEQUENTIAL SEEKS
004452  054462                          BNE     6$              ; IF SO, BRANCH
004453  102207  040000                  BIT     #INITW,R0       ; SEE IF INITIAL WRITE IN PROGRESS
004455  054462                          BNE     6$              ; IF SO, BRANCH
004456  101207  004000                  BIS     #NEWSUB,R0      ; FLAG AS TO GO ONTO A NEW SUBUNIT
004460  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE
004462  104207  000002          6$:     MOV     #NEWOP,R0       ; SET UP A NEW OPERATION
004464  114000  002217                  CLR     SCR1            ; CLEAR SCRATCH AREA1 (NO NEW SUBUNIT FLAG)
004466  104651  000023                  MOV     U.TSEC(R5),R1   ; GET TOTAL NUMBER OF SECTORS TO BE WRIITEN LAST OP
004470  054577                          BNE     4$              ; IF NOT FIRST TIME, BRANCH
004471  104200  177777  002217          MOV     #-1,SCR1        ; FLAG AS NEW SUBUNIT (WITHOUT GOING ONTO NEW SUBUNIT)
004474  004577                          BR      4$              ; EXIT
004475  103207  000110          5$:     BIC     #WCHECK!REDWRT,R0       ; CLEAR WRITE CHECK, MAKE OPERATION READ
004477  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE
004501  104657  000051                  MOV     U.MBN(R5),R0    ; GET LO LBN
004503  100657  000053                  MOV     R0,U.CBN(R5)    ; SAVE
004505  104657  000052                  MOV     U.MBN+1(R5),R0  ; GET HI LBN
004507  100657  000054                  MOV     R0,U.CBN+1(R5)  ; SAVE
004511  114002                          CLR     R2              ; NO ERRORS
004512  100652  000024                  MOV     R2,U.CSEC(R5)   ; SECTORS R/W IS ZERO
```

```
     90 004514  104657  000023          9$:     MOV     U.TSEC(R5),R0   ; GET TOTAL NUMBER OF SECTORS TO R/W
     91 004516  107657  000024                  SUB     U.CSEC(R5),R0   ; SUBTRACT HOW MANY DONE
     92 004520  104651  000046                  MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
     93 004522  102201  000200                  BIT     #RBNBN,R1       ; SEE IF AN RBN IS BEING HANDLED
     94 004524  014530                          BEQ     1$              ; IF NOT, BRANCH
     95 004525  104207  000001                  MOV     #1,R0           ; ONLY READ/WRITE ONE SECTOR
     96 004527  004546                          BR      8$              ; BRANCH
     97 004530  102201  000100          1$:     BIT     #REDWRT,R1      ; SEE IF WRITE IS IN PROGRESS
     98 004532  054541                          BNE     7$              ; IF SO, BRANCH
     99 004533  106307  002233                  CMP     SECMAX,R0       ; SEE IF TRYING TO READ MORE THAN POSSIBLE
    100 004535  044546                          BCC     8$              ; IF NOT, BRANCH
    101 004536  104307  002233                  MOV     SECMAX,R0       ; SET R0 TO MAXIMUM POSSIBLE SECTORS TO READ
    102 004540  004546                          BR      8$              ; BRANCH
    103 004541  106307  002234          7$:     CMP     CHNMAX,R0       ; SEE IF TRYING TO WRITE MORE THAN POSSIBLE
    104 004543  044546                          BCC     8$              ; IF NOT, BRANCH
    105 004544  104307  002234                  MOV     CHNMAX,R0       ; SET R0 TO MAXIMUM POSSIBLE SECTORS TO WRITE
    106 004546  100657  000022          8$:     MOV     R0,U.MSEC(R5)   ; SAVE IN NUMBER OF SECTORS TO R/W THIS PASS
    107 004550  104653  000031          3$:     MOV     U.MLEV(R5),R3   ; GET MAXIMUM ERROR REVCOVERY LEVEL
    108 004552  100653  000027                  MOV     R3,U.ELEV(R5)   ; STORE IN CURRENT LEVEL
    109 004554  104203  177777                  MOV     #-1,R3          ; START RETRIES AT 0
    110 004556  100653  000012                  MOV     R3,U.RWTO(R5)   ; SAVE IN UNIT PARAMETER AREA
    111                             ;           NOTE:   R2 IS ZERO AT THIS POINT
    112 004560  100652  000021                  MOV     R2,U.NSEC(R5)   ; NO SECTORES SUCCESSFULLY READ/WRITTEN
    113 004562  100652  000010                  MOV     R2,U.RRTY(R5)   ; ZERO READ RETRY VALUE
    114 004564  104207  000023                  MOV     #SEEK,R0        ; POINT TO SEEK AS NEXT OPERATION
    115 004566  024163                          CALL    ENABLE          ; ENABLE ERROR RECOVERY
    116 004567  102203  030000                  BIT     #LEVUSD!NXTLEV,R3       ; SEE IF ERROR RECOVERY LEVELS USED
    117 004571  014577                          BEQ     4$              ; IF NOT, BRANCH
    118 004572  024207                          CALL    GOSEEK          ; CAUSE A SEEK TO RESET ERROR RECOVERY
    119 004573  103203  030000                  BIC     #LEVUSD!NXTLEV,R3       ; NO ERROR RECOVERY IN PROGRESS
    120 004575  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
    121 004577  114001                  4$:     CLR     R1              ; IMMIDIATE CALL
    122 004600  003231                          BR      JMPRET          ; RETURN TO SEQUENCER
    123                                         .DSABL  LSB
    127         004602                  AREA1   =       .+1
```

```
                                .SBTTL  ********** OVERLAY MODULE NEWOP - SET UP NEW OPERATION (COMMON CODE TOO)
 004601                                 DMOVLY  NO,AREA1
 004601  000105                        .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
         000002                 NEWOP   =       SETUP+1
                                ;
                                ;       SET UP A NEW OPERATION
                                ;
                                        .ENABL  LSB
 004602  104147                        MOV     (R4),R0         ; GET SUBUNIT PARAMETERS
 004603                                 ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
 004603  102207  040100                BIT     #SEQSEK!INITW,R0        ; SEE IF BN'S ACCESSED SREQUENTIALLY
 004605  054617                        BNE     2$              ; IF SO, BRANCH
 004606  102207  000040                BIT     #BEUSED,R0      ; SEE IF BEGIN/END SETS USED
 004610  014614                        BEQ     1$              ; IF NOT, BRANCH
 004611  104207  000003                MOV     #RNDBE,R0       ; GET A RANDOM BLOCK NUMBER (BEGIN/END SETS)
 004613  004627                        BR      4$              ; EXIT
 004614  104207  000005        1$:     MOV     #RNDTG,R0       ; GET A RANDOM BLOCK NUMBER (TRACKS/GROUPS)
 004616  004627                        BR      4$              ; BRANCH
 004617  102207  000040        2$:     BIT     #BEUSED,R0      ; SEE IF BEGIN/END SETS USED
 004621  014625                        BEQ     3$              ; IF NOT, BRANCH
 004622  104207  000004                MOV     #SEQBE,R0       ; GET A SEQUENTIAL BLOCK NUMBER (BEGIN/END SETS)
 004624  004627                        BR      4$              ; EXIT (SHOULD BRANCH TO 7$) ********
 004625  104207  000006        3$:     MOV     #SEQTG,R0       ; GET A SEQUENTIAL BLOCK NUMBER (TRACKS/GROUPS)
 004627  114001                4$:     CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
 004630  114002                        CLR     R2              ; NO ERRORS
 004631  003231                        BR      JMPRET
                                        .DSABL  LSB
```

```
                                        .SBTTL  AFTOP  - AFTER THE SECTOR HAS BEEN DETERMINED, FIND OUT WHAT TO DO WITH IT
004632                                  AFTOP:
                                        ;
                                        ;       AFTER THE SECTORS TO BE READ/WRITTEN HAVE BEEN DETERMINED, SET UP THE
                                        ;       REST OF THE OPERATION
                                        ;
                                        .SBTTL  CLRUP  - CLEAR ALL PARAMETER BITS
                                        ;CLRUP
                                        ;
                                        ;       CLRUP CLEARS ALL NECESSARY FLAG BITS
                                        ;
004632  104657  000046                  ;       MOV     U.PARM(R5),R0   ; MOVE UNIT PARAMETERS TO R0
                                        ;       CLEAR ALL FLAGS BEFORE THE NEXT OPERATION
004634  103207  004712                  ;       BIC     #RBNBN!REVEC!WCHECK!DATCMP!REDWRT!NEWSUB,R0
004636  100657  000046                  ;       MOV     R0,U.PARM(R5)   ; SAVE UNIT PARAMETERS
```

```
                                .SBTTL  RORW   - DETERMINE IF A READ OR A WRITE IS TO BE DONE
                                ;RORW
                                ;
                                ;       RORW DETERMINES IF THE BLOCK SELECTED WILL BE READ OR WRITTEN
                                ;
 004640  104142                         MOV     (R4),R2          ; GET SUBUNIT PARAMETERS
 004641                                 ASSUME  S.PARM,0         ; ASSUME THAT S.PARM IS ZERO
 004641  102202  042000                 BIT     #INITW!WONLY,R2  ; SEE IF INITIAL WRITE OR WRITE ONLY
 004643  054652                         BNE     1$               ; IF SO, BRANCH
 004644  102202  004000                 BIT     #RONLY,R2        ; SEE IF READ ONLY
 004646  054656                         BNE     2$               ; IF SO, BRANCH
 004647  024764                         CALL    RANDOM           ; GET RANDOM NUMBER
 004650  110601                         ROR     R1               ; ROTATE LOW BIT INTO CARRY
 004651  044656                         BCC     2$               ; IF LOW BIT ZERO, BRANCH
 004652  101207  000100          1$:    BIS     #REDWRT,R0       ; SET READ OR WRITE BIT (WRITE)
 004654  100657  000046                 MOV     R0,U.PARM(R5)    ; SAVE UNIT PARAMETERS
 004656                          2$:
```

PATTRN - IF WRITE, DETERMINE WHAT PATTERN IS TO BE WRITTEN

```
                                        .SBTTL  PATTRN - IF WRITE, DETERMINE WHAT PATTERN IS TO BE WRITTEN
                                        ;PATTRN
                                        ;
                                        ;       FIND PATTERN TO USE (ONLY IF WRITE TO BE DONE)
                                        ;
004656  102207  000100                        BIT     #REDWRT,R0      ; SEE IF WRITE TO BE DONE
004660  014675                                BEQ     PATEXT          ; IF NOT, BRANCH
004661  104641  000004                        MOV     S.PAT(R4),R1    ; GET PATTERN NUMBER
004663  014667                                BEQ     RND0            ; IF PATTERN NOT SELECTED, BRANCH
004664  103201  177760                        BIC     #LBLONB,R1      ; SPECIAL PATTERN USED, CLEAR HI BITS (MAP 16 TO 0)
004666  004673                                BR      FIXPAT          ; IF PATTERN SELECTED, USE IT
004667  024764                  RND0:   CALL    RANDOM          ; GET RANDOM PATTERN NUMBER
004670  103201  177760                        BIC     #LBLONB,R1      ; CLEAR UNUSED BITS
004672  014667                                BEQ     RND0            ; NO SUCH THING AS PATTERN 0, TRY AGAIN
004673  100651  000014          FIXPAT: MOV     R1,U.PAT(R5)    ; SAVE THE PATTERN NUMBER
004675                          PATEXT:
```

```
                                        .SBTTL  WCHK   - IF WRITE, SEE IF A WRITE CHECK IS TO BE DONE
                                        ;WCHK
                                        ;
                                        ;       WCHK SETS THE WRITE CHECK BIT IF A WRITE CHECK IS TO BE DONE
                                        ;
 004675  102207  000100                         BIT     #REDWRT,R0      ; SEE IF WRITE IS TO BE DONE
 004677  014720                                 BEQ     2$              ; IF NOT, BRANCH
 004700  102202  000010                         BIT     #WCHECK,R2      ; SEE IF WRITE CHECK IS TO BE DONE
 004702  014720                                 BEQ     2$              ; IF NOT, BRANCH
 004703  102202  040000                         BIT     #INITW,R2       ; SEE IF INITIAL WRITE IS IN PROGRESS
 004705  054720                                 BNE     2$              ; IF SO, NO WRITE CHECK, SO BRANCH
 004706  102202  000004                         BIT     #WCHKAL,R2      ; SEE IF WRITE CHECK IS ALWAYS TO BE DONE
 004710  054714                                 BNE     1$              ; IF SO, BRANCH
 004711  024764                                 CALL    RANDOM          ; GET RANDOM NUMBER
 004712  110601                                 ROR     R1              ; 1/2 OF THE TIME WRITE CHECK
 004713  044720                                 BCC     2$              ; BRANCH (NO WRITE CHECK) IF LOWEST BIT CLEAR
 004714  101207  000010         1$:             BIS     #WCHECK,R0      ; SET WRITE CHECK BIT
 004716  100657  000046                         MOV     R0,U.PARM(R5)   ; SAVE UNIT PARAMETERS
 004720                         2$:
```

```
                                        .SBTTL  DCOMP  - IF READ OR WRITE W/WRITE CHECK, SEE IF DATA COMPARE IS TO BE DONE
                                        ;DCOMP
                                        ;
                                        ;       DCOMP SETS THE DATCMP BIT IF A DATA COMPARE IS TO BE DONE
                                        ;
 004720  102202  000002                         BIT     #DATCMP,R2      ; SEE IF DATA COMPARE REQUESTED
 004722  014743                                 BEQ     DCEXT           ; IF NOT, BRANCH
 004723  102207  000100                         BIT     #REDWRT,R0      ; SEE IF READ IS TO BE DONE
 004725  014731                                 BEQ     DCREAD          ; IF SO, BRANCH
 004726  102207  000010                         BIT     #WCHECK,R0      ; SEE IF WRITE CHECK IS TO BE DONE
 004730  014743                                 BEQ     DCEXT           ; IF NOT, BRANCH
 004731  102202  000001          DCREAD: BIT     #DCMPAL,R2      ; SEE IF DATA COMPARE ALWAYS DONE
 004733  054737                                 BNE     DCMPYS          ; IF SO, BRANCH
 004734  024764                                 CALL    RANDOM          ; GET RANDOM NUMBER
 004735  110601                                 ROR     R1              ; SEE IF DATA COMPARE SHOULD BE DONE
 004736  044743                                 BCC     DCEXT           ; IF NOT, BRANCH
 004737  101207  000002          DCMPYS: BIS     #DATCMP,R0      ; SET DATA COMPARE BIT
 004741  100657  000046                         MOV     R0,U.PARM(R5)   ; SAVE UNIT PARAMETERS
 004743                          DCEXT:
```

```
      1 004743  104657  000051                  MOV     U.MBN(R5),R0    ; GET LO LBN
      2 004745  100657  000053                  MOV     R0,U.CBN(R5)    ; SAVE
      3 004747  104657  000052                  MOV     U.MBN+1(R5),R0  ; GET HI LBN
      4 004751  100657  000054                  MOV     R0,U.CBN+1(R5)  ; SAVE
      5 004753  114002                          CLR     R2              ; NO ERRORS
      6 004754  100652  000024                  MOV     R2,U.CSEC(R5)   ; SECTORS R/W IS ZERO
      7 004756  100652  000021                  MOV     R2,U.NSEC(R5)   ; SECTORS R/W THIS PASS IS ZERO
      8 004760  114001                          CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
      9 004761  104207  000001                  MOV     #SETUP,R0       ; NEXT MODULE IS SETUP
     10 004763  003231                          BR      JMPRET          ; RETURN TO SEQUENCER
```

```
                                .SBTTL  RANDOM - RANDOM NUMBER ROUTINE
004764                          RANDOM:
                                ;
                                ;       RANDOM CALCULATES A RANDOM NUMBER AND RETURNS IT IN R1
                                ;       RANGE 0 - 2^16-1
                                ;
004764                                  PUSH    <R0,R2>         ; SAVE R0 AND R2
004764  100467                                                          MOV R0,-(SP)
004765  100462                                                          MOV R2,-(SP)
004766  104307  002224                 MOV     LOSEED,R0       ; MOVE LO ORDER SEED TO R0
004770  104301  002225                 MOV     HISEED,R1       ; MOVE HI SEED TO R1
004772  104202  000007                 MOV     #7,R2           ; MOVE LOOP COUNT TO R2
004774  110207                  1$:     ROL     R0              ; ROTATE LO ORDER NUMBER BY 1
004775  110201                          ROL     R1              ; ROTATE HI OREDR NUMBER BY 1 (PROPOGATE CARRY)
004776  103207  000001                 BIC     #1,R0           ; CLEAR LOWER BIT
005000  117402                         DEC     R2              ; DECREMENT COUNT
005001  054774                         BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
005002  105307  002224                 ADD     LOSEED,R0       ; ADD ORIGINAL SEED (X129)
005004  045006                         BCC     2$              ; IF NO CARRY, BRANCH
005005  115401                         INC     R1              ; PROPOGATE CARRY
005006  105301  002225          2$:     ADD     HISEED,R1       ; ADD HISEED
005010  105207  001057                 ADD     #1057,R0        ; ADD LO CONSTANT
005012  045014                         BCC     3$              ; IF NO CARRY, BRANCH
005013  115401                         INC     R1              ; PROPOGATE CARRY
005014  105201  047401          3$:     ADD     #47401,R1       ; ADD HI CONSTANT
005016  104070  002224                 MOV     R0,LOSEED       ; SAVE LO ORDER SEED
005020  104010  002225                 MOV     R1,HISEED       ; SAVE HI ORDER SEED
005022                                  POP     <R2,R0>         ; RESTORE R2 AND R0
005022  104262                                                          MOV (SP)+,R2
005023  104267                                                          MOV (SP)+,R0
005024  000000                          RETURN
```

```
                                .SBTTL  MASK    - FIND MASK FOR RANDOM NUMBER
005025                          MASK:
                                ;
                                ;       MASK MASKS OUT HIGHER BITS OF A RANDOM NUMBER, SO THE PROB. OF THE
                                ;       RANDOM NUMBER EXCEEDING THE MAXIMUM DESIRED IS LESSENED
                                ;
005025                          PUSH    R1              ; SAVE R1
005025  100461                                                          MOV R1,-(SP)
005026  114001                  CLR     R1              ; ZERO R1
005027  105011          1$:     ADD     R1,R1           ; ROTATE R1 ONE BIT LEFT
005030  101201  000001          BIS     #1,R1           ; TURN ON BIT 0
005032  106017                  CMP     R1,R0           ; SEE IF R1 IS GREATER THAN R0
005033                          BCS     1$              ; IF NOT, BRANCH
005033  045035                                                          BCC     .+2
005034  005027                                                          BR      1$
005035  104207  177777          MOV     #-1,R0          ; SET UP FOR COMPLEMENT
005037  103017                  BIC     R1,R0           ; COMPLEMENT R1
005040                          POP     R1              ; RESTORE R1
005040  104261                                                          MOV (SP)+,R1
005041  000000                  RETURN                  ; RETURN TO RNDLBN
        005043          AREA2   =       .+1
                                .IF     LE,MAXADR-.
                        MAXADR  =       .+1
                                .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE RNDBE - GET NEXT RANDOM BEGIN/END SET SECTOR
                                        DMOVLY  RB,AREA2
005042  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000003                          RNDBE   =       NEWOP+1
                                ;
                                ;       RNDBE GETS A RANDOM LBN NUMBER
                                ;
                                ;       GET A RANDOM BEGIN/END SET FROM THE SUBUNIT'S B/E SETS
                                ;
                                        .ENABL  LSB
005043  114007                          CLR     R0              ; INITILIZE COUNTER
005044  104203  000016                  MOV     #S.BESS+3,R3    ; R3 POINTS TO FIRST B/E SET
005046  105043                          ADD     R4,R3           ; R3 POINTS TO FIRST B/E SET
005047  104132                  CNTLOP: MOV     (R3),R2         ; MOVE EOL FLAG WORD TO R2
005050  075055                          BMI     COUNTD          ; IF END-OF-LIST, BRANCH
005051  105203  000004                  ADD     #4,R3           ; POINT TO NEXT EOL WORD
005053  115407                          INC     R0              ; INCREMENT COUNT
005054  005047                          BR      CNTLOP          ; IF NOT, TRY AGAIN
005055  115007                  COUNTD: TST     R0              ; SEE IF COUNT = 0
005056  015077                          BEQ     GOTOFS          ; IF SO, BRANCH
005057  024764                  TRYAGN: CALL    RANDOM          ; GET A RANDOM NUMBER
005060  103201  177774                  BIC     #177774,R1      ; MASK OUT ALL BUT 2 LOWEST BITS
005062  106207  000002                  CMP     #2,R0           ; TEST HOW COUNT RELATES TO 2
005064  075074                          BMI     GOTOBE          ; IF COUNT IS 3, BRANCH
005065  015071                          BEQ     THREBE          ; IF COUNT IS 2, BRANCH
005066  103201  177776                  BIC     #177776,R1      ; MASK OUT ALL BUT LOWEST BIT
005070  005074                          BR      GOTOBE          ; BRANCH
005071  106201  000002          THREBE: CMP     #2,R1           ; SEE IF RANDOM NUMBER OVER 2
005073  075057                          BMI     TRYAGN          ; IF SO, GET ANOTHER NUMBER
005074  104017                  GOTOBE: MOV     R1,R0           ; MOVE RANDOM NUMBER TO R0
005075  105077                          ADD     R0,R0           ; R0 X 2
005076  105077                          ADD     R0,R0           ; R0 X 2
005077  105207  000013          GOTOFS: ADD     #S.BESS,R0      ; POINT TO RANDOM BEGIN/END SET
005101  105047                          ADD     R4,R0           ; POINT TO RANDOM BEGIN/END SET
```

```
                        ;
                        ;       ONCE A RANDOM B/E SET HAS BEEN FOUND, FIND A RANDOM LBN
                        ;       WITHIN THAT B/E SET
                        ;
005102                  PUSH    <R4,R5>         ; SAVE R4 AND R5
005102  100464                                                         MOV R4,-(SP)
005103  100465                                                         MOV R5,-(SP)
005104  104274          MOV     (R0)+,R4        ; R4 GETS LO ORDER ENDING LBN
005105  104275          MOV     (R0)+,R5        ; R5 GETS HI ORDER ENDING LBN
005106  107274          SUB     (R0)+,R4        ; SUBTRACT LO ORDER STARTING LBN
005107  045111          BCC     NOBORO          ; IF NO BORROW, BRANCH
005110  117405          DEC     R5              ; ADJUST FOR BORROW
005111  104171  NOBORO: MOV     (R0),R1         ; GET HI WORD
005112  103201  170000  BIC     #170000,R1      ; STRIP OFF EOL FLAG (IF ANY)
005114  107015          SUB     R1,R5           ; SUBTRACT HI ORDER STARTING LBN
005115  117407          DEC     R0              ; POINT TO LO ORDER STARTING LBN
005116                  PUSH    R0              ; SAVE R0
005116  100467                                                         MOV R0,-(SP)
005117  024764  10$:    CALL    RANDOM          ; GET A RANDOM NUMBER
005120  114003          CLR     R3              ;
005121  104301  002224  MOV     LOSEED,R1       ; ORIGINAL SEED IN R1
005123  115005          TST     R5              ; SEE IF R5 IS ZERO
005124  015136          BEQ     11$             ; IF SO, BRANCH
005125  104057          MOV     R5,R0           ; ELSE, STORE IN R0
005126  025025          CALL    MASK            ; COMPUTE RANDOM NUMBER MASK
005127  104303  002225  MOV     HISEED,R3       ; MOVE RANDOM NUMBER INTO R3
005131  103073          BIC     R0,R3           ; CLEAR WITH MASK
005132  106053          CMP     R5,R3           ; CMP MAXIMUM RANDOM NUM TO RANDOM NUM
005133                  BCS     10$             ; IF TOO LARGE, DO AGAIN
005133  045135                                                         BCC     .+2
005134  005117                                                         BR      10$
005135  055144          BNE     CLCLBN          ; IF NOT THE SAME, BRANCH
005136  104047  11$:    MOV     R4,R0           ; MOVE R4 TO R0 TO COMPUTE MASK
005137  025025          CALL    MASK            ; COMPUTE MASK
005140  103071          BIC     R0,R1           ; MASK OUT HIGH BITS
005141  106041          CMP     R4,R1           ; SEE IF MAX RND NUM < RANDOM NUMBER
005142                  BCS     10$             ; IF SO, DO AGAIN
005142  045144                                                         BCC     .+2
005143  005117                                                         BR      10$
005144  104012  CLCLBN: MOV     R1,R2           ; MOVE RANDOM NUMBER TO R2
005145                  POP     R0              ; RESTORE R0
005145  104267                                                         MOV (SP)+,R0
005146  105272          ADD     (R0)+,R2        ; ADD LO ORDER STARTING BLOCK TO RANDOM NUM
005147  045151          BCC     3$              ; IF NO CARRY, BRANCH
005150  115403          INC     R3              ; ADJUST HIGH ORDER FOR CARRY
005151  104171  3$:     MOV     (R0),R1         ; MOVE HI ORDER LBN TO R1
005152  103201  170000  BIC     #170000,R1      ; STRIP OFF EOL FLAG IF ANY
005154  105013          ADD     R1,R3           ; ADD HI ORDER LBN
005155                  POP     <R5,R4>         ; RESTORE R5, R4
005155  104265                                                         MOV (SP)+,R5
005156  104264                                                         MOV (SP)+,R4
005157  107207  000003  SUB     #3,R0           ; R0 POINTS TO ENDING LBN
005161  104271          MOV     (R0)+,R1        ; R1 HAS LO ORDER ENDING LBN
005162  104177          MOV     (R0),R0         ; R0 HAS HI ORDER ENDING LBN
005163  107021          SUB     R0,R1           ; SUBTRACT LO LBN FROM LO ENDING LBN
005164  075171          BMI     4$              ; IF RESULT IS NEGATIVE, BRANCH
005165  045167          BCC     1$              ; IF NO CARRY, BRANCH
```

```
     48 005166  117407                        DEC     R0              ; DECREMENT HI ENDING LBN
     49 005167  107037                1$:     SUB     R3,R0           ; SUBTRACT HI LBN FROM HI ENDING LBN
     50 005170  015173                        BEQ     2$              ; IF HI WORD IS ZERO, BRANCH
     51 005171  104201  077776        4$:     MOV     #77776,R1       ; MOVE MAXIMUM COUNT TO R1
     52 005173  115401                2$:     INC     R1              ; INCREMENT MAXIMUM SECTOR COUNT
     53 005174  104017                        MOV     R1,R0           ; COPY TO R0 FOR MAXMUM
     54
     55 005175                                PUSH    R3              ; SAVE R3
        005175  100463                                                                  MOV R3,-(SP)
     56 005176  024764                        CALL    RANDOM          ; GET A RANDOM NUMBER OF SECTORS TO READ/WRITE
     57 005177  103201  177774                BIC     #MAXMSK,R1      ; SET READ/WRITE LIMIT
     58 005201  115401                        INC     R1              ; MAKE MAXIMUM OF LIMIT+1
     59 005202  106071                        CMP     R0,R1           ; SEE IF RANDOM NUMBER EXCEEDS POSSIBLE
     60 005203  075205                        BMI     6$              ; IF SO, BRANCH
     61 005204  104017                        MOV     R1,R0           ; ONLY READ/WRITE THE RANDOM NUMBER OF SECTORS
     62 005205  100657  000023        6$:     MOV     R0,U.TSEC(R5)   ; SAVE IN TSEC
     63 005207                                POP     R3              ; RESTORE R3
        005207  104263                                                                  MOV (SP)+,R3
     64 005210  104207  000051                MOV     #U.MBN,R0       ; R0 POINTS TO LBN AREA
     65 005212  105057                        ADD     R5,R0           ; R0 POINTS TO LBN AREA
     66 005213  100272                        MOV     R2,(R0)+        ; MOVE LO ORDER TO LBN AREA
     67 005214  100173                        MOV     R3,(R0)         ; MOVE HI ORDER TO LBN AREA
     68 005215  104207  004632                MOV     #AFTOP,R0       ; NEXT MODULE IS AFTOP
     69 005217  114001                        CLR     R1              ; IMMIDATE CALL
     70 005220  114002                        CLR     R2              ; NO ERRORS
     71 005221  003231                        BR      JMPRET
     72                                       .DSABL  LSB
     76                                       .IF     LE,MAXADR-.
     77                               MAXADR  =       .+1
     78                                       .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE SEQBE - GET NEXT SEQUENTIAL BEGIN/END SET SECTOR
                                        DMOVLY  SB,AREA2
005222  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000004                          SEQBE   =       RNDBE+1
                                ;
                                ;       SEQBE FINDS THE NEXT SEQUENTIAL LBN
                                ;
                                        .ENABL  LSB
005043  104657  000046                  MOV     U.PARM(R5),R0   ; MOVE UNIT PARAMETERS TO R0
005045  102207  010000                  BIT     #DIREC,R0       ; TEST DIRECTION
005047  015154                          BEQ     UP              ; IF SEQUENTIAL UP, E ANCH
005050  115000  002217                  TST     SCR1            ; SEE IF NEW SUBUNIT
005052  015075                          BEQ     5$              ; IF NOT, BRANCH
005053  104201  000013                  MOV     #S.BESS,R1      ; R0 WILL POINT TO BEGIN/END SETS
005055  105041                          ADD     R4,R1           ; R0 POINTS TC BEGIN/END SETS
005056  104617  000003          6$:     MOV     3(R1),R0        ; SEE IF END-OF-LIST
005060  075064                          BMI     8$              ; IF SO, BRANCH
005061  105201  000004                  ADD     #4,R1           ; POINT TO NEXT BEGIN/END SET
005063  005056                          BR      6$              ; LOOP
005064  104202  000051          8$:     MOV     #U.MBN,R2       ; R2 WILL POINT TO MASTER BN
005066  105052                          ADD     R5,R2           ; R2 POINTS TO MASTER BN
005067  104217                          MOV     (R1)+,R0        ; GET LO ORDER ENDING SET
005070  100127                          MOV     R0,(R2)         ; SAVE
005071  104217                          MOV     (R1)+,R0        ; GET HI ORDER ENDING SET
005072  100627  000001                  MOV     R0,1(R2)        ; SAVE
005074  005114                          BR      7$              ; BRANCH
005075  025266                  5$:     CALL    FNDBES          ; FIND BEGIN/END SET THAT LBN IS IN
005076  105201  000002                  ADD     #2,R1           ; POINT TO BEGIN SET
005100  021765                          CALL    CMP2            ; SEE IF LBN = BEGINNING LBN
005101  015134                          BEQ     BESDWN          ; IF SO, BRANCH
005102  104127                          MOV     (R2),R0         ; MOVE LOW ORDER WORD TO R0
005103  107207  000001                  SUB     #1,R0           ; DECREMENT R0
005105  100127                          MOV     R0,(R2)         ; SAVE R0
005106  045114                          BCC     7$              ; IF NO BORROW, BRANCH
005107  104627  000001                  MOV     1(R2),R0        ; MOVE HIGH ORDER WORD TO R0
005111  117407                          DEC     R0              ; DECREMENT R0
005112  100627  000001                  MOV     R0,1(R2)        ; SAVE R0
005114  104227                  7$:     MOV     (R2)+,R0        ; MOVE LO ORDER BN TO R0
005115  104223                          MOV     (R2)+,R3        ; MOVE HI ORDER BN TO R3
005116  107217                          SUB     (R1)+,R0        ; SUBTRACT BEGIN BN FROM BN
005117  075127                          BMI     4$              ; IF RESULT IS NEGATIVE, BRANCH
005120  045122                          BCC     1$              ; IF NO BORROW, BRANCH
005121  117403                          DEC     R3              ; PROPOGATE BORROW
005122  104111                  1$:     MOV     (R1),R1         ; GET BEGINNING BN
005123  103201  170000                  BIC     #^CHBHINB,R1    ; CLEAR EOL FLAG, IF ANY
005125  107013                          SUB     R1,R3           ; SUBTRACT HI ORDER BN FROM HI BN
005126  015131                          BEQ     2$              ; IF EQUAL, BRANCH
005127  104207  077776          4$:     MOV     #77776,R0       ; MOVE MAXIMUM COUNT TO R0
005131  115407                  2$:     INC     R0              ; MAKE SUBTRACTION INCLUSIVE
005132  025305                  3$:     CALL    MAXMUM          ; SET MAXIMUM COUNT (U.TSEC AND U.MSEC)
```

```
     74 005133  005135                        BR      NLBEXT          ; BRANCH
     75                                       .DSABL  LSB
     76 005134  025325                BESDWN: CALL    PREVBE          ; FIND PREVIOUS B/E SET OR UNIT
     77 005135  104651  000023        NLBEXT: MOV     U.TSEC(R5),R1   ; GET TOTAL NUMBER OF SECTORES TO READ/WRITE
     78 005137  117401                        DEC     R1              ; DECREMENT SECTOR COUNT
     79 005140  104653  000051                MOV     U.MBN(R5),R3    ; GET LO ORDER MASTER BN
     80 005142  107013                        SUB     R1,R3           ; SUBTRACT SECTORS TO READ/WRITE
     81 005143  100653  000051                MOV     R3,U.MBN(R5)    ; SAVE
     82 005145  045153                        BCC     1$              ; IF NO CARRY, BRANCH
     83 005146  104653  000052                MOV     U.MBN+1(R5),R3  ; GET HI ORDER MASTER BN
     84 005150  117403                        DEC     R3              ; PROPOGATE CARRY
     85 005151  100653  000052                MOV     R3,U.MBN+1(R5)  ; SAVE
     86 005153  005251                1$:     BR      BEEXT           ; BRANCH
     87 005154  115000  002217        UP:     TST     SCR1            ; SEE IF FIRST TIME ON THIS SUBUNIT
     88 005156  015177                        BEQ     5$              ; IF NOT, BRANCH
     89 005157  104201  000013                MOV     #S.BESS,R1      ; R0 WILL POINT TO BEGIN/END SETS
     90 005161  105041                        ADD     R4,R1           ; R0 POINTS TO BEGIN/END SETS
     91 005162  104202  000051                MOV     #U.MBN,R2       ; R2 WILL POINT TO MASTER BN
     92 005164  105052                        ADD     R5,R2           ; R2 POINTS TO MASTER BN
     93 005165  104617  000002                MOV     2(R1),R0        ; GET LO ORDER STARTING SET
     94 005167  100127                        MOV     R0,(R2)         ; SAVE
     95 005170  104617  000003                MOV     3(R1),R0        ; GET HI ORDER STARTING SET
     96 005172  103207  170000                BIC     #^CHBHINB,R0    ; CLEAR EOL FLAG IF ANY
     97 005174  100627  000001                MOV     R0,1(R2)        ; SAVE
     98 005176  005233                        BR      7$              ; BRANCH
     99 005177                        5$:
    100                               .SBTTL  UPLBN  - IF GOING UP, UPDATE CURRENT BN TO LAST BN READ/WRITTEN
    101                               ;UPLBN
    102                               ;
    103                               ;       UPLBN UPDATES U.MBN TO POINT TO THE LAST LBN READ/WRITTEN
    104                               ;       (INCREASING LBN)
    105                               ;
    106 005177                                PUSH    <R0,R2>         ; SAVE R0,R2
        005177  100467                                                                  MOV R0,-(SP)
        005200  100462                                                                  MOV R2,-(SP)
    107 005201  104202  000051                MOV     #U.MBN,R2       ; R2 POINTS TO LAST LBN
    108 005203  105052                        ADD     R5,R2           ; R2 POINTS TO LAST LBN
    109 005204  104657  000023                MOV     U.TSEC(R5),R0   ; GET NUMBER OF SECTORS WRITTEN
    110 005206  117407                        DEC     R0              ; SUBTRACT ONE
    111 005207  105127                        ADD     (R2),R0         ; ADD LOW ORDER LBN TO R0
    112 005210  100227                        MOV     R0,(R2)+        ; SAVE LOW ORDER WORD, POINT TO HIGH ORDER
    113 005211  045215                        BCC     8$              ; IF NO CARRY, BRANCH
    114 005212  104127                        MOV     (R2),R0         ; GET HIGH ORDER WORD
    115 005213  115407                        INC     R0              ; INCREMENT
    116 005214  100127                        MOV     R0,(R2)         ; SAVE HIGH ORDER WORD
    117 005215                        8$:     POP     <R2,R0>         ; RESTORE R0,R2
        005215  104262                                                                  MOV (SP)+,R2
        005216  104267                                                                  MOV (SP)+,R0
    118 005217  025266                        CALL    FNDBES          ; FIND BEGIN/END SET THAT LBN IS IN
    119 005220  015250                        BEQ     BESUP           ; IF LBN = ENDING LBN, BRANCH
    120 005221  104127                        MOV     (R2),R0         ; MOVE LOW ORDER WORD TO R0
    121 005222  105207  000001                ADD     #1,R0           ; INCREMENT R0
    122 005224  100127                        MOV     R0,(R2)         ; SAVE R0
    123 005225  045233                        BCC     7$              ; IF NO CARRY, BRANCH
    124 005226  104627  000001                MOV     1(R2),R0        ; MOVE HIGH ORDER WORD TO R0
    125 005230  115407                        INC     R0              ; INCREMENT R0
    126 005231  100627  000001                MOV     R0,1(R2)        ; SAVE R0
```

UPLBN - IF GOING UP, UPDATE CURRENT BN TO LAST BN READ/WRITTE

```
127 005233  104217                7$:     MOV     (R1)+,R0        ; R0 HAS LO ENDING LBN
128 005234  104113                        MOV     (R1),R3         ; R3 HAS HI ENDING LBN
129 005235  107227                        SUB     (R2)+,R0        ; SUBTRACT LO LBN FROM LO ENDING LBN
130 005236  075243                        BMI     4$              ; IF RESULT IS NEGATIVE, BRANCH
131 005237  045241                        BCC     1$              ; IF NO CARRY, BRANCH
132 005240  117403                        DEC     R3              ; PROPOGATE CARRY
133 005241  107123                1$:     SUB     (R2),R3         ; SUBTRACT HI LBN FROM ENDING LBN
134 005242  015245                        BEQ     2$              ; IF ZERO, BRANCH
135 005243  104207  077776        4$:     MOV     #77776,R0       ; MOVE MAXIMUM COUNT TO R0
136 005245  115407                2$:     INC     R0              ; MAKE SUBTRACTION INCLUSIVE
137 005246  025305                3$:     CALL    MAXMUM          ; SET MAXIMUM COUNT (U.TSEC AND U.MSEC)
138 005247  005251                        BR      BEEXT           ; BRANCH
139 005250  025361                BESUP:  CALL    NEXTBE          ; FIND NEXT B/E SET OR UNIT
140 005251  104657  000046        BEEXT:  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
141 005253  102207  004000                BIT     #NEWSUB,R0      ; SEE IF NEW SUBUNIT
142 005255  015261                        BEQ     1$              ; IF NOT, BRANCH
143 005256  104207  000002                MOV     #NEWOP,R0       ; NEWOP NEXT MODULE
144 005260  005263                        BR      2$              ; BRANCH
145 005261  104207  004632        1$:     MOV     #AFTOP,R0       ; NEXT MODULE IS AFTOP
146 005263  114001                2$:     CLR     R1              ; IMMIDATE CALL
147 005264  114002                        CLR     R2              ; NO ERRORS
148 005265  003231                        BR      JMPRET          ; RETURN TO SEQUENCER
```

FNDBES - FIND THE BEGIN/END SET CURRENT BN RESIDES IN

```
      1                                  .SBTTL  FNDBES - FIND THE BEGIN/END SET CURRENT BN RESIDES IN
      2 005266                           FNDBES:
      3                                  ;
      4                                  ;       FNDBES FINDS THE BEGIN/END SET THAT THE CURRENT LBN RESIDES IN
      5                                  ;
      6 005266  104203  177777                   MOV     #-1,R3              ; START COUNTER (ADJUSTED)
      7 005270  104201  000007                   MOV     #S.BESS-4,R1        ; POINT TO B/E SETS
      8 005272  105041                           ADD     R4,R1               ; POINT TO B/E SETS
      9 005273  104202  000051                   MOV     #U.MBN,R2           ; POINT TO LBN
     10 005275  105052                           ADD     R5,R2               ; POINT TO LBN
     11 005276  105201  000004           FNDLOP: ADD     #4,R1               ; INCREMENT B/E SET POINTER
     12 005300  115403                           INC     R3                  ; INCREMENT COUNT
     13 005301  021765                           CALL    CMP2                ; COMPARE LBN WITH ENDING LBN
     14 005302  015304                           BEQ     OUTO                ; IF = TO, BRANCH
     15 005303  045276                           BCC     FNDLOP              ; IF LBN > ENDING LBN, BRANCH
     16 005304  000000           OUTO:   RETURN                              ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  MAXMUM - FIND HOW MANY SECTORS TO READ/WRITE
005305                                  MAXMUM:
                                        ;
                                        ;       MAXMUM WILL WRITE THE MAXIMUM NUMBER OF SECTORS IF IN INITIAL WRITE,
                                        ;       OTHERWISE IT WILL R/W ONE TRACK
                                        ;
005305  115000  002223                  TST     M.PARM          ; SEE IF INITIAL WRITE IS IN PROGRESS
005307  075322                          BMI     1$              ; IF SO, BRANCH
005310                                  ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
005310  115000  002217                  TST     SCR1            ; SEE IF FIRST TIME ON THIS UNIT
005312  015315                          BEQ     2$              ; IF NOT, BRANCH
005313  025433                          CALL    NXTTRK          ; ALIGN WITH NEXT TRACK
005314  005324                          BR      3$              ; EXIT
005315  106647  000006          2$:     CMP     S.TRKL(R4),R0   ; SEE IF TRACK GREATER THAN MAX
005317  035322                          BPL     1$              ; IF LESS THAN TRACK LENGTH, BRANCH
005320  104647  000006                  MOV     S.TRKL(R4),R0   ; R/W ONE TRACK
005322  100657  000023          1$:     MOV     R0,U.TSEC(R5)   ; SAVE NUMBER OF SECTORS TO R/W
005324  000000                  3$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  PREVBE - MOVE TO PREVIOUS BEGIN/END SET
005325                          PREVBE:
                                ;
                                ;       PREVBE IS CALLED IF THE LBN'S ARE DECREASING, AND THE LAST LBN
                                ;       WAS THE FIRST LBN WITHIN A B/E SET.  PREVBE FINDS THE PREVIOUS
                                ;       B/E SET, OR, IF IT IS THE LAST ON THE SUBUNIT, FINDS THE LAST B/E
                                ;       SET OF THE PREVIOUS UNIT.  IF IT IS THE FIRST BE SET OF THE FIRST
                                ;       UNIT, THE DIRECTION OF THE LBN'S ARE REVERSED (THEY BEGIN GOING UP
                                ;       THE UNIT)
                                ;
005325  115003                          TST     R3              ; SEE IF ALLREADY ON 1ST B/E SET
005326  055336                          BNE     5$              ; IF NOT, BRANCH
005327  104657  000046                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005331  101207  004000                  BIS     #NEWSUB,R0      ; MARK AS GOING ON TO THE NEXT UNIT
005333  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE
005335  005360                          BR      6$              ; EXIT
005336  107201  000006          5$:     SUB     #6,R1           ; POINT TO PREVIOUS ENDING SET
005340  025420                          CALL    MOV2            ; MOVE ENDING SET TO LBN
005341  104217                          MOV     (R1)+,R0        ; R0 HAS LO ENDING LBN
005342  104212                          MOV     (R1)+,R2        ; R2 HAS HI ENDING LBN
005343  107217                          SUB     (R1)+,R0        ; SUBTRACT LO BEGINNING LBN FROM LO ENDING
005344  075354                          BMI     4$              ; IF RESULT IS NEGATIVE, BRANCH
005345  045347                          BCC     1$              ; IF NO CARRY, BRANCH
005346  117402                          DEC     R2              ; PROPOGATE CARRY
005347  104111          1$:             MOV     (R1),R1         ; GET BEGINNING BN
005350  103201  170000                  BIC     #^CHBHINB,R1    ; CLEAR EOL FLAG, IF ANY
005352  107012                          SUB     R1,R2           ; SUBTRACT HI ORDER BN FROM HI BN
005353  015356                          BEQ     2$              ; IF ZERO, BRANCH
005354  104207  077776          4$:     MOV     #77776,R0       ; MOVE MAXIMUM COUNT TO R0
005356  115407          2$:             INC     R0              ; TO MAKE IT AN INCLUSIVE SUBTRACT
005357  025433          3$:             CALL    NXTTRK          ; SET MAXIMUM COUNT (U.TSEC AND U.MSEC)
005360  000000          6$:             RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  NEXTBE - MOVE TO NEXT BEGIN/END SET
005361                                  NEXTBE:
                                        ;
                                        ;       NEXTBE IS CALLED WHEN THE CURRENT LBN IS THE LAST OF ITS B/E SET.
                                        ;       NEXTBE GOES TO THE NEXT B/E SET FOR THE NEXT LBN.  IF THE LBN IS
                                        ;       THE LAST ON THE SUBUNIT, IT GOES TO THE FIRST B/E SET OF THE NEXT
                                        ;       UNIT.  IF IT IS THE LAST B/E SET ON THE LAST UNIT, THE DIRECTION IS
                                        ;       CHANGED (DOWN), AND THE INITIAL WRITE BIT IS CLEARED IN CASE AN
                                        ;       INITIAL WRITE IS IN  PROGRESS
                                        ;
005361  104617  000003                          MOV     3(R1),R0        ; SEE IF THIS IS LAST B/E SET ON THIS UNIT
005363  035373                                  BPL     1$              ; IF SO, BRANCH
005364  104657  000046                          MOV     U.PARM(R5),R0   ; GET SUBUNIT PARAMETERS
005366  101207  004000                          BIS     #NEWSUB,R0      ; MARK AS GOING ON TO A NEW SUBUNIT
005370  100657  000046                          MOV     R0,U.PARM(R5)   ; SAVE
005372  005417                                  BR      7$              ; EXIT
005373  105201  000006          1$:             ADD     #6,R1           ; POINT TO NEXT BEGINNING LBN
005375  025420                                  CALL    MOV2            ; MOVE BEGINNING LBN TO LBN
005376  107201  000002                          SUB     #2,R1           ; R1 POINTS TO END SET
005400  104217                                  MOV     (R1)+,R0        ; R0 HAS LO ENDING LBN
005401  104213                                  MOV     (R1)+,R3        ; R3 HAS HI ENDING SET
005402  107217                                  SUB     (R1)+,R0        ; SUBTRACT BEGINNING LBN
005403  075413                                  BMI     5$              ; IF RESULT IS NEGATIVE, BRANCH
005404  045406                                  BCC     2$              ; IF NO CARRY, BRANCH
005405  117403                                  DEC     R3              ; PROPOGATE CARRY
005406  104111                  2$:             MOV     (R1),R1         ; GET BEGINNING BN
005407  103201  170000                          BIC     #^CHBHINB,R1    ; CLEAR EOL FLAG, IF ANY
005411  107013                                  SUB     R1,R3           ; SUBTRACT HI ORDER BN FROM HI BN
005412  015415                                  BEQ     3$              ; IF ZERO, BRANCH
005413  104207  077776          5$:             MOV     #77776,R0       ; MOVE MAXIMUM COUNT TO R0
005415  115407                  3$:             INC     R0              ; INCREMENT R0 (INCLUSIVE SUBTRACT)
005416  025433                  4$:             CALL    NXTTRK          ; SET MAXIMUM COUNT (U.TSEC AND U.MSEC)
005417  000000                  7$:             RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  MOV2   - 28 BIT MOVE
005420                          MOV2:
                                ;
                                ;       MOV2 COPIES A 28 BIT NUMBER POINTED TO BY R1 TO TWO WORDS POINTED
                                ;       TO BY R2.  BITS 31-28 ARE CLEARED BEFORE COPYING
                                ;
005420                          PUSH    R0              ; SAVE R0
005420  100467                                                  MOV R0,-(SP)
005421  104117                  MOV     (R1),R0         ; MOVE SOURCE LOW ORDER WORD TO R0
005422  100127                  MOV     R0,(R2)         ; MOVE R0 TO DESTINATION LOW ORDER WORD
005423  104617  000001          MOV     1(R1),R0        ; MOVE SOURCE HIGH ORDER WORD TO R0
005425  103207  170000          BIC     #^CHBHINB,R0    ; STRIP OFF UNUSED BITS
005427  100627  000001          MOV     R0,1(R2)        ; MOVE R0 TO DESTINATION HIGH ORDER WORD
005431                          POP     R0              ; RESTORE R0
005431  104267                                                  MOV (SP)+,R0
005432  000000                  RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  NXTTRK - FIND HOW MANY SECTORS TO R/W TO ALIGN WITH NEXT TRACK BOUNDRY
005433                          NXTTRK:
                                ;
                                ;       COMPUTE HOW MANY SECTORS TO READ/WRITE TO GET ON A TRACK BOUNDRY
                                ;       THIS WILL ALLOW SEQUENTIAL READS AND WRITES TO BE PERFORMED
                                ;       WITHOUT ANY 'BACKWARDS' SEEKING (EXCEPT IN THE CASE OF REVECTORS)
                                ;
005433                                  PUSH    R0              ; SAVE R0
005433  100467                                                                  MOV R0,-(SP)
005434  115000  002223                       TST     M.PARM          ; SEE IF INITIAL WRITE
005436  075553                           BMI     7$              ; IF SO, EXIT (WRITE MAXIMUM POSSIBLE)
005437                                   ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
005437  104651  000051                   MOV     U.MBN(R5),R1    ; GET LO ORDER SECTOR TO R/W
005441  104010  002175                   MOV     R1,CURBN        ; MOVE TO CALC AREA
005443  104651  000052                   MOV     U.MBN+1(R5),R1  ; GET HI ORDER SECTOR TO R/W
005445  104010  002176                   MOV     R1,CURBN+1      ; MOVE TO CALC AREA
005447  114007                           CLR     R0              ; TELL CALC TO SETUP ALL REQUIRED PARAMETERS
005450  022027                           CALL    CALC            ; CALC TRACK THAT SECTOR IS ON
005451  104207  005575                   MOV     #CYLSCR,R0      ; R0 POINTS TO SCRATCH SPACE
005453  104201  002201                   MOV     #CYL,R1         ; R1 POINTS TO CYL, GRP AND TRACK
005455  104202  000004                   MOV     #4,R2           ; MOVE 4 WORDS
005457  104213                   8$:     MOV     (R1)+,R3        ; GET WORD
005460  100273                           MOV     R3,(R0)+        ; SAVE
005461  117402                           DEC     R2              ; DECREMENT COUNT
005462  055457                           BNE     8$              ; IF INCOMPLETE, BRANCH
005463  104207  000020                   MOV     #20,R0          ; START INCREMENT BY 10 (16 DECIMAL)
005465  104651  000046           1$:     MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
005467  025556                           CALL    ID              ; INCREMENT OR DECREMENT (DEPENDING ON DIRECTION)
005470                                   PUSH    R0              ; SAVE R0
005470  100467                                                                  MOV R0,-(SP)
005471  022027                           CALL    CALC            ; CALCULATE NEW CYL/GRP/TRK
005472  104207  005575                   MOV     #CYLSCR,R0      ; R0 POINTS TO SCRATCH SPACE
005474  104201  002201                   MOV     #CYL,R1         ; R1 POINTS TO CYL, GRP AND TRACK
005476  104202  000004                   MOV     #4,R2           ; MOVE 4 WORDS
005500  104213                   9$:     MOV     (R1)+,R3        ; GET WORD
005501  106273                           CMP     (R0)+,R3        ; COMPARE
005502  055512                           BNE     2$              ; IF CHANGE, BRANCH
005503  117402                           DEC     R2              ; DECREMENT COUNT
005504  055500                           BNE     9$              ; IF INCOMPLETE, BRANCH
005505                                   POP     R0              ; RESTORE STEPSIZE
005505  104267                                                                  MOV (SP)+,R0
005506  106207  000020                   CMP     #20,R0          ; SEE IF ORIGINAL STEPSIZE
005510  015465                           BEQ     1$              ; IF SO, BRANCH
005511  005523                           BR      3$              ; BRANCH
005512                           2$:     POP     R0              ; RESTORE STEPSIZE
005512  104267                                                                  MOV (SP)+,R0
005513  106207  000001                   CMP     #1,R0           ; SEE IF LAST STEPSIZE
005515  015531                           BEQ     4$              ; FOUND FIRST SECTOR ON ANOTHER TRACK
005516  104201  177777                   MOV     #-1,R1          ; SET UP FOR COMPLEMENT
005520  103651  000046                   BIC     U.PARM(R5),R1   ; COMPLEMENT
005522  025556                           CALL    ID
005523  110607                   3$:     ROR     R0              ; ROTATE TO HALVE STEPSIZE
005524  103207  177760                   BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
005526  055465                           BNE     1$              ; IF NON-ZERO, LOOP
005527  115407                           INC     R0              ; MAKE R3 1
005530  005465                           BR      1$              ; LOOP
005531  104653  000046           4$:     MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
```

```
                        005533  102203  010000                  BIT     #DIREC,R3       ; FIND WHAT DIRECTION YOU'RE GOING IN
                        005535  015543                          BEQ     5$              ; IF UP, BRANCH
                        005536  104653  000051                  MOV     U.MBN(R5),R3    ; GET LO ORDER STARTING BN
                        005540  107303  002175                  SUB     CURBN,R3        ; SUBTRACT ENDING
                        005542  005547                          BR      6$              ; GO WITH COMPUTED NUMBER
                        005543  104303  002175          5$:     MOV     CURBN,R3        ; GET LO ORDER ENDING BN
                        005545  107653  000051                  SUB     U.MBN(R5),R3    ; SUBTRACT STARTING BN
                        005547                          6$:     POP     R0              ; GET MAX SECTORS THAT CAN BE R/W
                        005547  104267                                                                  MOV (SP)+,R0
                        005550  106073                          CMP     R0,R3           ; COMPARE
                        005551  075553                          BMI     7$              ; IF CAN WRITE MORE THAN POSSIBLE, BRANCH
                        005552  104037                          MOV     R3,R0           ; SETUP TO WRITE TO EOT
                        005553  100657  000023          7$:     MOV     R0,U.TSEC(R5)   ; SAVE IN SECTORS TO R/W
                        005555  000000                          RETURN                  ; RETURN TO CALLING PROGRAM

                        005556                          ID:
                        005556  102201  010000                  BIT     #DIREC,R1       ; SEE IF GOING UP OR DOWN
                        005560  015567                          BEQ     1$              ; IF UP, BRANCH
                        005561  107070  002175                  SUB     R0,CURBN        ; DECREMENT BN
                        005563  045574                          BCC     2$              ; IF NO BORROW, EXIT
                        005564  117400  002176                  DEC     CURBN+1         ; PROPOGATE BORROW
                        005566  005574                          BR      2$              ; EXIT
                        005567  105070  002175          1$:     ADD     R0,CURBN        ; INCREMENT BN
                        005571  045574                          BCC     2$              ; IF NO CARRY, BRANCH
                        005572  115400  002176                  INC     CURBN+1         ; PROPOGATE CARRY
                        005574  000000                  2$:     RETURN

                        005575  000000          CYLSCR: .WORD   0               ; SCRATCH AREA FOR STORING CYL, GRP AND TRK
                        005576  000000                  .WORD   0
                        005577  000000                  .WORD   0
                        005600  000000                  .WORD   0
                                                        .IF     LE,MAXADR-.
                                                MAXADR  =       .+1
                                                        .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE RNDTG - RANDOMLY GET NEXT TRACK/GROUP SECTOR
005601                                  DMOVLY  RT,AREA2
005601  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
000005                          RNDTG   =       SEQBE+1
                                ;
                                ;       COME UP WITH A RANDOM BN TO TEST USING THE TRACK/GROUP SETS
                                ;
                                ;       FIRST COME UP WITH A RANDOM COUNT <= MAXIMUM COUNT
                                ;
                                        .ENABL  LSB
005043  024764                  1$:     CALL    RANDOM          ; GET RANDOM NUMBER
005044  114002                          CLR     R2              ; HI ORDE WORD IS ZERO
005045  104301  002224                  MOV     LOSEED,R1       ; LOSEED IN R1
005047  104647  000014                  MOV     S.MCNT+1(R4),R0 ; GET MAXIMUM HI ORDER WORD
005051  015063                          BEQ     3$              ; IF ZERO, BRANCH
005052  025025                          CALL    MASK            ; CREATE MASK FOR RANDOM NUMBER
005053  104302  002225                  MOV     HISEED,R2       ; PUT NUMBER IN R2
005055  103072                          BIC     R0,R2           ; STRIP OFF UNUSED BITS
005056  106642  000014                  CMP     S.MCNT+1(R4),R2 ; SEE IF MAX EXCEEDED
005060                                  BCS     1$              ; IF SO, TRY AGAIN
005060  045062                                                                  BCC     .+2
005061  005043                                                                  BR      1$
005062  055073                          BNE     5$              ; IF NOT EQUAL, EXIT
005063  104647  000013          3$:     MOV     S.MCNT(R4),R0   ; GET LO ORDER MAX
005065  025025                          CALL    MASK            ; GET RANDOM MASK
005066  103071                          BIC     R0,R1           ; STRIP OFF BITS
005067  106641  000013                  CMP     S.MCNT(R4),R1   ; CHECK WITH MAX
005071                                  BCS     1$              ; IF GREATER THAN MAX, DO AGAIN
005071  045073                                                                  BCC     .+2
005072  005043                                                                  BR      1$
```

```
                                        ;
                                        ;       NOW FIND THE BN ASSOCIATED WITH THAT COUNT
                                        ;
005073                          5$:     PUSH    <R5,R4>         ; PUSH THE UNIT AND SUBUNIT POINTERS
005073  100465                                                                 MOV R5,-(SP)
005074  100464                                                                 MOV R4,-(SP)
005075  104205  000015                 MOV     #S.TGOF,R5      ; R5 WILL POINT TO THE TRACK/GROUP OFFSETS
005077  105045                         ADD     R4,R5           ; R5 POINTS TO THE TRACK/GROUP ORIGINAL OFFSET
005100  104257                         MOV     (R5)+,R0        ; GET LO ORDER OFFSET, MOVE TO RUNNING TOTAL
005101  104253                         MOV     (R5)+,R3        ; GET HI ORDER OFFSET, MOVE TO RUNNING TOTAL
005102  107201  000001         6$:     SUB     #1,R1           ; DECREMENT LO COUNT BY ONE
005104  045111                         BCC     7$              ; IF NO BORROW, BRANCH
005105  107202  000001                 SUB     #1,R2           ; DECREMENT HI COUNT BY ONE
005107                                  BCS     9$              ; IF COUNT EXHAUSTED, BRANCH
005107  045111                                                                 BCC     .+2
005110  005123                                                                 BR      9$
005111  104254                 7$:     MOV     (R5)+,R4        ; GET T/G OFFSET
005112  055117                         BNE     8$              ; IF NOT END OF LIST, BRANCH
005113  104205  000017                 MOV     #S.TGSS,R5      ; R5 WILL POINT TO START OF T/G LIST
005115  105165                         ADD     (SP),R5         ; R5 POINTS TO START OF T/G LIST
005116  104254                         MOV     (R5)+,R4        ; GET T/G OFFSET
005117  105047                 8$:     ADD     R4,R0           ; ADD TO RUNNING TOTAL
005120  045102                         BCC     6$              ; IF NO CARRY, BRANCH
005121  115403                         INC     R3              ; PROPOGATE CARRY
005122  005102                         BR      6$              ; BRANCH
005123                          9$:     POP     <R4,R5>         ; RESTORE
005123  104264                                                                 MOV (SP)+,R4
005124  104265                                                                 MOV (SP)+,R5
```

```
                                        ;
                                        ;       NOW FIND A RANDOM OFFSET INTO THE RANDOM TRACK OR GROUP
                                        ;
005125                                  PUSH    R0              ; STORE LO ORDER TOTAL ON STACK
005125  100467                                                                  MOV R0,-(SP)
005126  104642  000006                  MOV     S.TRKL(R4),R2   ; R2 HAS TRACK LENGTH
005130  104147                          MOV     (R4),R0         ; GET SUBUNIT PARAMETERS
005131                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
005131  102207  000020                  BIT     #TRACKS,R0      ; SEE IF USING TRACKS
005133  055147                          BNE     11$             ; IF SO, BRANCH
005134  104641  000007                  MOV     S.SCHR(R4),R1   ; R1 POINTS TO SUBUNIT PARAMETERS
005136  104611  000003                  MOV     TRKGRP(R1),R1   ; R1 HAS NUMBER OF TRACKS/GROUP
005140  103201  177400                  BIC     #HIBYTE,R1      ; CLEAR UNUSED BITS
005142  114007                          CLR     R0              ; CLEAR RUNNING TOTAL
005143  105027                  10$:    ADD     R2,R0           ; FIND SECTORS/GROUP
005144  117401                          DEC     R1              ; DECREMENT COUNT
005145  055143                          BNE     10$             ; IF UNEXHAUSTED, BRANCH
005146  104072                          MOV     R0,R2           ; MOVE COUNT TO R2 (MATCH WITH TRACKS ABOVE)
005147                          11$:    PUSH    R2              ; SAVE ON STACK
005147  100462                                                                  MOV R2,-(SP)
005150  117402                          DEC     R2              ; ADJUST COUNT
005151  104027                          MOV     R2,R0           ; MOVE TO MASKING REGISTER
005152  025025                          CALL    MASK            ; FIND MASK
005153  024764                  12$:    CALL    RANDOM          ; GET RANDOM NUMBER
005154  103071                          BIC     R0,R1           ; STRIP OFF UNUSED BITS
005155  106021                          CMP     R2,R1           ; SEE IF RANDOM NUMBER SMALL ENOUGH
005156                                  BCS     12$             ; IF NOT, BRANCH
005156  045160                                                                  BCC     .+2
005157  005153                                                                  BR      12$
005160                                  POP     R2              ; GET NUMBER OF SECTORS
005160  104262                                                                  MOV (SP)+,R2
005161  107012                          SUB     R1,R2           ; R2 IS NOW SECTORS REMAINING IN T/G
005162  105261                                                                  ADD     (SP)+,R1
005163  100651  000051                  MOV     R1,U.MBN(R5)    ; SAVE
005165  045167                          BCC     13$             ; IF NO CARRY, BRANCH
005166  115403                          INC     R3              ; PROPOGATE CARRY
005167  100653  000052          13$:    MOV     R3,U.MBN+1(R5)  ; SAVE HI ORDER STARTING BN
```

```
                                        ;
                                        ;       NOW COME UP WITH A RANDOM NUMBER OF SECTORS TO R/W WITHOUT GOING
                                        ;       OFF THE TRACK/GROUP THAT WE'RE ON
                                        ;
005171  106202  000016                 CMP     #16,R2          ; ONLY WRITE A MAXIMUM OF 16 SECTORS
005173  035177                         BPL     15$             ; IF LESS THAN 16, BRANCH
005174  104207  177774                 MOV     #MAXMSK,R0      ; SET READ/WRITE LIMIT
005176  005201                         BR      14$             ; BRANCH
005177  104027                 15$:    MOV     R2,R0           ; R0 IS SECTORS LEFT ON THIS T/G
005200  025025                         CALL    MASK            ; GET MASK
005201  024764                 14$:    CALL    RANDOM          ; GET RANDOM NUMBER
005202  103071                         BIC     R0,R1           ; STRIP OFF UNUSED BITS
005203  115401                         INC     R1              ; MUST BE GREATER THAN ZERO
005204  106021                         CMP     R2,R1           ; SEE IF SMALL ENOUGH
005205                                 BCS     14$             ; IF NOT, BRANCH
005205  045207                                                                 BCC     .+2
005206  005201                                                                 BR      14$
005207  100651  000023                 MOV     R1,U.TSEC(R5)   ; SAVE SECTORS TO WRITE
005211  104207  004632                 MOV     #AFTOP,R0       ; NEXT MODULE IS AFTOP
005213  114001                         CLR     R1              ; IMMIDATE CALL
005214  114002                         CLR     R2              ; NO ERRORS
005215  003231                         BR      JMPRET          ; RETURN TO SEQNCR
                                       .DSABL   LSB
                                       .IF      LE,MAXADR-.
                                MAXADR =        .+1
                                       .ENDC
```

```
                                        .SBTTL  ********** OVERLAY MODULE SEQTG - GET NEXT SEQUENTIAL TRACK/GROUP SECTOR
                                                DMOVLY  ST,AREA2
005216  000105                                  .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;
                                        ;
                                        ;
000006                                  SEQTG   =       RNDTG+1
                                        ;
                                        ;       FIND THE NEXT TRACK/GROUP TO TEST SEQUENTIALLY
                                        ;
                                                .ENABL  LSB
005043  104657  000046                          MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005045  104141                          1$:     MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
005046                                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
005046  102207  010000                          BIT     #DIREC,R0       ; SEE IF GOING UP (INITIAL WRITE) OR DOWN
005050  015060                                  BEQ     3$              ; IF GOING UP OR INITIAL WRITE, BRANCH
005051  102201  000020                          BIT     #TRACKS,R1      ; SEE IF TRACKS OR GROUPS
005053  015056                                  BEQ     2$              ; IF GROUPS, BRANCH
005054  025200                                  CALL    DOWNT           ; GO DOWN A TRACK
005055  005066                                  BR      5$              ; CALCULATE NUMBER OF SECTORS TO READ/WRITE
005056  025342                          2$:     CALL    DOWNG           ; GO DOWN A GROUP
005057  005066                                  BR      5$              ; CALCULATE NUMBER OF SECTORS TO READ/WRITE
005060  102201  000020                  3$:     BIT     #TRACKS,R1      ; SEE IF TRACKS OR GROUPS
005062  015065                                  BEQ     4$              ; IF GROUPS, BRANCH
005063  025121                                  CALL    UPT             ; GO UP A TRACK
005064  005066                                  BR      5$              ; CALCULATE NUMBER OF SECTORS TO READ/WRITE
005065  025262                          4$:     CALL    UPG             ; GO UP A GROUP
005066  104642  000006                  5$:     MOV     S.TRKL(R4),R2   ; GET SECTORS/TRACK
005070  104141                                  MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
005071                                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
005071  115000  002223                          TST     M.PARM          ; SEE IF INITIAL WRITE IN PROGRESS
005073  035112                                  BPL     7$              ; IF NOT, BRANCH
005074                                          ASSUME  IWIPRG,100000   ; ASSUME IWIPRG IS SIGN BIT
005074  102201  000020                          BIT     #TRACKS,R1      ; SEE IF TRACKS IN USE
005076  055112                                  BNE     7$              ; IF SO, BRANCH
005077  104027                                  MOV     R2,R0           ; COPY SECTORS/TRACK TO R0
005100  104641  000007                          MOV     S.SCHR(R4),R1   ; GET POINTER TO SUBUNIT CHARACTERISTICS
005102  104611  000003                          MOV     TRKGRP(R1),R1   ; GET TRACKS/GROUP
005104  103201  177400                          BIC     #HIBYTE,R1      ; CLEAR UNUSED BITS
005106  114002                                  CLR     R2              ; CLEAR RUNNING TOTAL
005107  105072                          6$:     ADD     R0,R2           ; ADD TO RUNNING TOTAL
005110  117401                                  DEC     R1              ; DECREMENT COUNT
005111  055107                                  BNE     6$              ; IF INCOMPLETE, BRANCH
005112  100652  000023                  7$:     MOV     R2,U.TSEC(R5)   ; SAVE NUMBER OF SECTORS TO PROCESS
005114  104207  004632                  8$:     MOV     #AFTOP,R0       ; NEXT MODULE IS AFTOP
005116  114001                                  CLR     R1              ; IMMIDATE CALL
005117  114002                                  CLR     R2              ; NO ERRORS
005120  003231                                  BR      JMPRET          ; RETURN TO SEQNCR
                                                .DSABL  LSB
```

```
                                        .SBTTL  UPT     - MOVE UP ONE TRACK
005121                                  UPT:
                                        ;
                                        ;       POINT U.MBN TO THE NEXT TRACK TO TEST (IT DOESN'T HAVE TO BE A
                                        ;       TRACK, BUT IF GROUPS, U.MBN MUST BE ADJUSTED BACK TO THE FIRST
                                        ;       BN IN THE GROUP
                                        ;
005121  115000  002217                          TST     SCR1            ; SEE IF NEW SUBUNIT
005123  015126                                 BEQ     10$             ; IF NOT, BRANCH
005124  025376                                 CALL    NEWUPT          ; SETUP NEW SUBUNIT
005125  005177                                 BR      4$              ; EXIT
005126  104657  000015          10$:    MOV     U.CCNT(R5),R0   ; GET LO ORDER COUNT
005130  107207  000001                  SUB     #1,R0           ; DECREMENT
005132  100657  000015                  MOV     R0,U.CCNT(R5)   ; SAVE
005134  045153                          BCC     2$              ; IF NO BORROW, BRANCH
005135  104657  000016                  MOV     U.CCNT+1(R5),R0 ; GET HI ORDER COUNT
005137  107207  000001                  SUB     #1,R0           ; PROPOGATE BORROW
005141  045151                          BCC     1$              ; IF NO BORROW, BRANCH
005142  104657  000046                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005144  101207  004000                  BIS     #NEWSUB,R0      ; FLAG AS TO GO ONTO NEXT SUBUNIT
005146  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE
005150  005177                          BR      4$              ; EXIT
005151  100657  000016          1$:     MOV     R0,U.CCNT+1(R5) ; SAVE
005153  104657  000017          2$:     MOV     U.PCTG(R5),R0   ; GET POINTER TO CURRENT TRACK/GROUP
005155  104271                          MOV     (R0)+,R1        ; GET OFFSET TO NEXT TRACK
005156  055163                          BNE     3$              ; IF OFFSET, BRANCH
005157  104207  000017                  MOV     #S.TGSS,R0      ; R0 WILL POINT TO START OF OFFSETS
005161  105047                          ADD     R4,R0           ; R0 POINTS TO OFFSETS
005162  104271                          MOV     (R0)+,R1        ; R1 HAS OFFSET
005163  100657  000017          3$:     MOV     R0,U.PCTG(R5)   ; SAVE POINTER TO CURRENT TRACK/GROUP
005165  105651  000051                  ADD     U.MBN(R5),R1    ; ADD OFFSET TO CURRENT BN
005167  100651  000051                  MOV     R1,U.MBN(R5)    ; SAVE
005171  045177                          BCC     4$              ; IF NO CARRY, EXIT
005172  104651  000052                  MOV     U.MBN+1(R5),R1  ; GET HI ORDER BN
005174  115401                          INC     R1              ; PROPOGATE CARRY
005175  100651  000052                  MOV     R1,U.MBN+1(R5)  ; SAVE
005177  000000                  4$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  DOWNT  - MOVE DOWN ONE TRACK
005200                                  DOWNT:
                                        ;
                                        ;       MOVE THE MASTER BN DOWN TO TEST THE PROCEEDING TRACK/GROUP
                                        ;
005200  115000  002217                  TST     SCR1            ; SEE IF NEW SUBUNIT
005202  015205                         BEQ     10$             ; IF NOT, BRANCH
005203  025424                         CALL    NEWDNT          ; INITIALIZE THIS SUBUNIT
005204  005261                         BR      5$              ; EXIT
005205  104657  000015          10$:    MOV     U.CCNT(R5),R0   ; GET LO ORDER COUNT
005207  107207  000001                  SUB     #1,R0           ; DECREMENT COUNT
005211  100657  000015                  MOV     R0,U.CCNT(R5)   ; SAVE
005213  045232                         BCC     2$              ; IF NO BORROW, BRANCH
005214  104657  000016                  MOV     U.CCNT+1(R5),R0 ; GET HI ORDER COUNT
005216  107207  000001                  SUB     #1,R0           ; PROPOGATE BORROW
005220  045230                         BCC     1$              ; IF NO CARRY, BRANCH
005221  104657  000046                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005223  101207  004000                  BIS     #NEWSUB,R0      ; FLAG AS TO GO ONTO NEXT SUBUNIT
005225  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE
005227  005261                         BR      5$              ; EXIT
005230  100657  000016          1$:     MOV     R0,U.CCNT+1(R5) ; SAVE
005232  104201  000017          2$:     MOV     #S.TGSS,R1      ; R0 WILL POINT TO START OF T/G OFFSETS
005234  105041                         ADD     R4,R1           ; R0 POINTS TO START OF T/G OFFSETS
005235  104657  000017                  MOV     U.PCTG(R5),R0   ; GET POINTER INTO LIST
005237  106071                         CMP     R0,R1           ; SEE IF AT START OF LIST
005240  055244                         BNE     4$              ; IF NOT, BRANCH
005241  115407          3$:     INC     R0              ; R0 POINTS TO NEXT WORD
005242  104171                         MOV     (R0),R1         ; SEE IF END OF LIST
005243  055241                         BNE     3$              ; IF NOT, BRANCH
005244  104651  000051          4$:     MOV     U.MBN(R5),R1    ; GET LO ORDER MASTER BN
005246  107471                         SUB     -(R0),R1        ; SUBTRACT OFFSET
005247  100651  000051                  MOV     R1,U.MBN(R5)    ; SAVE
005251  100657  000017                  MOV     R0,U.PCTG(R5)   ; SAVE POINTER
005253  045261                         BCC     5$              ; IF NO BORROW, EXIT
005254  104657  000052                  MOV     U.MBN+1(R5),R0  ; GET HI ORDER WORD
005256  117407                         DEC     R0              ; PROPOGATE BORROW
005257  100657  000052                  MOV     R0,U.MBN+1(R5)  ; SAVE
005261  000000          5$:     RETURN                          ; RETURN TO CALLING PROGRAM
```

```
UPG    - MOVE UP ONE GROUP

                                .SBTTL  UPG    - MOVE UP ONE GROUP
005262                          UPG:
                                ;
                                ;       MOVE MASTER BN UP TO THE NEXT GROUP
                                ;
005262  115000  002217          10$:    TST     SCR1            ; SEE IF THIS IS A NEW SUBUNIT
005264  015270                          BEQ     10$             ; IF NOT, BRANCH
005265  025376                          CALL    NEWUPT          ; INITILIZE THE SUBUNIT FOR TRACKS
005266  025500                          CALL    SETSEC          ; INITILIZE THE SUBUNIT FOR GROUPS
005267  005341                          BR      5$              ; EXIT
005270  102201  040000                  BIT     #INITW,R1       ; SEE IF AN INITIAL WRITE IS BEING DONE
005272  055337                          BNE     4$              ; IF SO, BRANCH
005273  104641  000006                  MOV     S.TRKL(R4),R1   ; R1 HAS TRACK LENGTH
005275  104657  000020                  MOV     U.CTRK(R5),R0   ; GET TRACK COUNT
005277  117407                          DEC     R0              ; DECREMENT COUNT
005300  015316                          BEQ     1$              ; IF COUNT EXHAUSTED, BRANCH
005301  100657  000020                  MOV     R0,U.CTRK(R5)   ; SAVE
005303  105651  000051                  ADD     U.MBN(R5),R1    ; MOVE BN TO NEXT TRACK
005305  100651  000051                  MOV     R1,U.MBN(R5)    ; SAVE
005307  045341                          BCC     5$              ; EXIT
005310  104657  000052                  MOV     U.MBN+1(R5),R0  ; GET HI ORDER BN
005312  115407                          INC     R0              ; PROPOGATE CARRY
005313  100657  000052                  MOV     R0,U.MBN+1(R5)  ; SAVE
005315  005341                          BR      5$              ; EXIT
005316  025500                  1$:     CALL    SETSEC          ; SETUP NEXT TRACK/GROUP COUNT
005317  114002                          CLR     R2              ; CLEAR RUNNING TOTAL
005320  117407                  2$:     DEC     R0              ; DECREMENT COUNT
005321  015324                          BEQ     3$              ; IF COUNT EXPIRED, BRANCH
005322  105012                          ADD     R1,R2           ; ADD SECTORS/TRACK TO RUNNING TOTAL
005323  005320                          BR      2$              ; LOOP
005324  104657  000051          3$:     MOV     U.MBN(R5),R0    ; GET LO ORDER MASTER BN
005326  107027                          SUB     R2,R0           ; ADJUST BACK TO START OF GROUP
005327  100657  000051                  MOV     R0,U.MBN(R5)    ; SAVE
005331  045337                          BCC     4$              ; IF NO BORROW, BRANCH
005332  104657  000052                  MOV     U.MBN+1(R5),R0  ; GET HI ORDER MASTER BN
005334  117407                          DEC     R0              ; PROPOGATE BORROW
005335  100657  000052                  MOV     R0,U.MBN+1(R5)  ; SAVE
005337  114007                  4$:     CLR     R0              ; CLEAR R0 SO UPT DOESN'T DETECT A NEW SUBUNIT
005340  025121                          CALL    UPT             ; GO TO NEXT GROUP
005341  000000                  5$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  DOWNG  - MOVE DOWN ONE GROUP
005342                                  DOWNG:
                                        ;
                                        ;       MOVE MASTER BN TO THE NEXT LOWER GROUP
                                        ;
005342  115000  002217                  TST     SCR1            ; SEE IF THIS IS A NEW SUBIUNIT
005344  015350                          BEQ     10$             ; IF NOT, BRANCH
005345  025424                          CALL    NEWDNT          ; SETUP THIS SUBUNIT FOR TRACKS
005346  025511                          CALL    LSTTRK          ; SETUP THIS SUBUNIT FOR GROUPS
005347  005375                          BR      4$              ; EXIT
005350  104657  000020          10$:    MOV     U.CTRK(R5),R0   ; GET TRACK COUNT
005352  117407                          DEC     R0              ; DECREMENT COUNT
005353  015373                          BEQ     1$              ; IF EXPIRED, BRANCH
005354  100657  000020                  MOV     R0,U.CTRK(R5)   ; SAVE
005356  104657  000051                  MOV     U.MBN(R5),R0    ; GET LO ORDER MASTER BN
005360  107647  000006                  SUB     S.TRKL(R4),R0   ; SUBTRACT TRACK LENGTH
005362  100657  000051                  MOV     R0,U.MBN(R5)    ; SAVE
005364  045375                          BCC     4$              ; EXIT
005365  104657  000052                  MOV     U.MBN+1(R5),R0  ; GET HI MASTER BN
005367  117407                          DEC     R0              ; PROPOGATE BORROW
005370  100657  000052                  MOV     R0,U.MBN+1(R5)  ; SAVE
005372  005375                          BR      4$              ; EXIT
005373  025200                  1$:     CALL    DOWNT           ; GO DOWN A GROUP
005374  025511                          CALL    LSTTRK          ; SET MASTER BN ON LAST TRACK OF GROUP
005375  000000                  4$:     RETURN                  ; RETURN TO CALLING PROGRM
```

```
                                        .SBTTL  NEWUPT - INITILIZE PARAMETERS FOR SEQUENTIALLY UP BY TRACKS
005376                                  NEWUPT:
                                        ;
                                        ;       INITIALIZE THIS SUBUNIT FOR SEQUENTIALLY ACCESSING TRACKS IN
                                        ;       ASCENDING ORDER
                                        ;
005376  104647  000013                        MOV     S.MCNT(R4),R0   ; GET MAXIMUM COUNT
005400  100657  000015                        MOV     R0,U.CCNT(R5)   ; SAVE IN CURRENT COUNT
005402  104647  000014                        MOV     S.MCNT+1(R4),R0 ; GET HI ORDER MAXIMUM COUNT
005404  100657  000016                        MOV     R0,U.CCNT+1(R5) ; SAVE IN HI ORDER CURRENT COUNT
005406  104647  000015                        MOV     S.TGOF(R4),R0   ; GET STARTING OFFSET
005410  100657  000051                        MOV     R0,U.MBN(R5)    ; SAVE IN MASTER BN
005412  104647  000016                        MOV     S.TGOF+1(R4),R0 ; GET HI ORDER STARTING OFFSET
005414  100657  000052                        MOV     R0,U.MBN+1(R5)  ; SAVE IN HI ORDER MASTER BN
005416  104207  000017                        MOV     #S.TGSS,R0      ; R0 WILL POINT TO START OF OFFSETS
005420  105047                              ADD     R4,R0           ; R0 POINTS TO START OF OFFSETS
005421  100657  000017                        MOV     R0,U.PCTG(R5)   ; SAVE IN POINTER TO CURRENT TRACK/GROUP
005423  000000                              RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  NEWDNT - INITILIZE PARAMETERS FOR SEQUENTIALLY DOWN BY TRACKS
005424                          NEWDNT:
                                ;
                                ;       INITIALIZE THIS SUBUNIT FOR SEQUENTIALLY ACCESSING TRACKS IN
                                ;       DESCENDING ORDER
                                ;
005424                                  PUSH    <R5,R4>         ; SAVE UNIT AND SUBUNIT POINTERS
005424  100465                                                                  MOV R5,-(SP)
005425  100464                                                                  MOV R4,-(SP)
005426  104642  000015                  MOV     S.TGOF(R4),R2   ; R2 HAS LO ORDER STARTING OFFSET
005430  104643  000016                  MOV     S.TGOF+1(R4),R3 ; R3 HAS HI ORDER STARTING OFFSET
005432  104647  000013                  MOV     S.MCNT(R4),R0   ; R0 HAS LO ORDER MAXIMUM COUNT
005434  100657  000015                  MOV     R0,U.CCNT(R5)   ; SAVE LO ORDER MAXIMUM COUNT
005436  104641  000014                  MOV     S.MCNT+1(R4),R1 ; R1 HAS HI ORDER MAXIMUM COUNT
005440  100651  000016                  MOV     R1,U.CCNT+1(R5) ; SAVE HI ORDER MAXIMUM COUNT
005442  105204  000017                  ADD     #S.TGSS,R4      ; R4 POINTS TO START OF OFFSETS
005444  107207  000001          1$:     SUB     #1,R0           ; DECREMENT COUNT
005446  045453                          BCC     2$              ; IF NO CARRY, BRANCH
005447  107201  000001                  SUB     #1,R1           ; PROPOGATE BORROW
005451                                  BCS     4$              ; IF COUNT EXHAUSTED, BRANCH
005451  045453                                                                  BCC     .+2
005452  005465                                                                  BR      4$
005453  104245          2$:             MOV     (R4)+,R5        ; GET OFFSET
005454  055461                          BNE     3$              ; IF NOT END-OF-LIST, BRANCH
005455  104204  000017                  MOV     #S.TGSS,R4      ; R4 WILL POINT TO START OF OFFSETS
005457  105164                          ADD     (SP),R4         ; R4 POINTS TO START OF OFFSETS
005460  104245                          MOV     (R4)+,R5        ; GET OFFSET
005461  105052          3$:             ADD     R5,R2           ; ADD OFFSET TO BN
005462  045444                          BCC     1$              ; IF NO CARRY, LOOP
005463  115403                          INC     R3              ; PROPOGATE CARRY
005464  005444                          BR      1$              ; LOOP
005465  104665  000001          4$:     MOV     1(SP),R5        ; GET UNIT POINTER
005467  100654  000017                  MOV     R4,U.PCTG(R5)   ; SAVE POINTER TO CURRENT TRACK/GROUP
005471                                  POP     <R4,R5>         ; RESTORE
005471  104264                                                                  MOV (SP)+,R4
005472  104265                                                                  MOV (SP)+,R5
005473  100652  000051                  MOV     R2,U.MBN(R5)    ; SAVE LO ORDER STARTING BN
005475  100653  000052                  MOV     R3,U.MBN+1(R5)  ; SAVE HI ORDER
005477  000000                          RETURN                  ; RETURN TO CALLING PROGRAM
```

```
      1                                     .SBTTL  SETSEC - SETUP TRACK/GROUP COUNT FOR SELECTED GROUPS
      2 005500                              SETSEC:
      3                                     ;
      4                                     ;       MOVE TRACKS/GROUP TO U.CTRK
      5                                     ;
      6 005500  104647  000007                      MOV     S.SCHR(R4),R0   ; R0 POINTS TO SUBUNIT CHARACTERISTICS
      7 005502  104677  000003                      MOV     TRKGRP(R0),R0   ; R0 HAS TRACKS/GROUPS
      8 005504  103207  177400                      BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
      9 005506  100657  000020                      MOV     R0,U.CTRK(R5)   ; SAVE AS TRACK COUNT
     10 005510  000000                              RETURN                  ; RETURN TO CALLING MODULE
```

```
                                        .SBTTL  LSTTRK - SET MASTER BN TO POINT TO LAST TRACK IN THE GROUP
005511                                  LSTTRK:
                                        ;
                                        ;       SET MBN TO LAST TRACK OF GROUP
                                        ;
005511  025500                          CALL    SETSEC          ; SETUP TRACK/GROUP COUNT
005512  114002                          CLR     R2              ; CLEAR RUNNING TOTAL
005513  117407                  2$:     DEC     R0              ; DECREMENT COUNT
005514  015520                          BEQ     3$              ; IF COUNT EXPIRED, BRANCH
005515  105642  000006                  ADD     S.TRKL(R4),R2   ; ADD SECTORS/TRACK TO RUNNING LENGTH
005517  005513                          BR      2$              ; LOOP
005520  105652  000051          3$:     ADD     U.MBN(R5),R2    ; MBN WILL POINT TO LAST TRACK IN GROUP
005522  100652  000051                  MOV     R2,U.MBN(R5)    ; SAVE
005524  045530                          BCC     4$              ; IF NO CARRY, EXIT
005525  104657  000052                  MOV     U.MBN+1(R5),R0  ; GET HI ORDER BN
005527  115407                          INC     R0              ; PROPOGATE CARRY
005530  100657  000052          4$:     MOV     R0,U.MBN+1(R5)  ; SAVE
005532  000000                          RETURN                  ; RETURN TO CALLING MODULE
                                        .IF     LE,MAXADR-.
                                MAXADR  =       .+1
                                        .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE BUILDP - BUILD THE READ/WRITE LINKED LISTS
                                        DMOVLY  BP,AREA0
005533  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
000007                          BUILDP  =       SEQTG+1         ; READ/WRITE LINKED LIST BUILDER
                                ;
                                ;       BUILD THE READ/WRITE LINKED LISTS
                                ;
                                        .ENABL  LSB
004413  021031                          CALL    TLKHST          ; COMMUNICATE WITH THE HOST SO NOT TO TIME OUT
004414  104657  000046                  MOV     U.PARM(R5),R0   ; R0 HAS UNIT PARAMETERS
004416  104650  000022  002235          MOV     U.MSEC(R5),MAXSEC       ; GET NUMBER OF SECTORS TO READ/WRITE
004421  102207  000200                  BIT     #RBNBN,R0       ; SEE IF RBN TO BE READ/WRITTEN
004423  014433                          BEQ     1$              ; IF NOT, BRANCH
004424  104650  000055  002175          MOV     U.RBN(R5),CURBN ; MOVE LO ORDER TO CURRENT BN
004427  104650  000056  002176          MOV     U.RBN+1(R5),CURBN+1     ; MOVE HI ORDER TO CURRENT BN
004432  004441                          BR      4$              ; BRANCH
004433  104650  000053  002175  1$:     MOV     U.CBN(R5),CURBN         ; MOVE LO ORDER TO CURRENT BN
004436  104650  000054  002176          MOV     U.CBN+1(R5),CURBN+1     ; MOVE LO ORDER TO CURRENT BN
004441  104300  002227  004701  4$:     MOV     MEMPOL,TMEMPL   ; START OF BUILD (INITILIZE MEM POOL)
004444  114007                          CLR     R0              ; FORCE THE CALCULATION TABLE TO BE FILLED
004445  104070  002232                  MOV     R0,CHAINS       ; MARK CHAIN AS EMPTY
004447  022027                  5$:     CALL    CALC            ; CALCULATE CYL, TRK AND GRP
004450  106650  000064  002201          CMP     U.CCYL(R5),CYL  ; SEE IF LO CYL MATCHES
004453  054654                          BNE     7$              ; IF NOT, BRANCH
004454  106650  000065  002202          CMP     U.CCYL+1(R5),CYL+1      ; SEE IF HI CYL MATCHES
004457  054654                          BNE     7$              ; IF NOT, BRANCH
004460  106650  000066  002203          CMP     U.CGRP(R5),GROUP        ; SEE IF GROUP MATCHES
004463  054654                          BNE     7$              ; IF NOT, BRANCH
```

LINK - BUILD A LINK (NODE) IN THE READ/WRITE CHAIN

```
                                        .SBTTL  LINK    - BUILD A LINK (NODE) IN THE READ/WRITE CHAIN
                                        ;LINK
                                        ;
                                        ;       LINK WILL CREATE THE LINKS FOR THE CHAIN
                                        ;
 004464  104653  000046                        MOV     U.PARM(R5),R3   ; R3 HAS UNIT PARAMETERS
 004466  104307  002232                        MOV     CHAINS,R0       ; R0 POINTS TO START OF CHAIN
 004470  054512                                BNE     22$             ; IF THE CHAIN IS ALLREADY STARTED, BRANCH
 004471  102203  000100                        BIT     #REDWRT,R3      ; SEE IF READ IN PROGRESS
 004473  014504                                BEQ     21$             ; IF SO, BRANCH
 004474  102203  000400                        BIT     #REVEC,R3       ; SEE IF TRYING TO FIND A REPLACEMENT
 004476  054504                                BNE     21$             ; IF SO, YOU'RE TRYING TO READ, SO BRANCH
 004477  104207  000401                        MOV     #WBUFLN,R0      ; R0 HAS NUMBER OF WORDS IN WRITE BUFFER
 004501  024674                                CALL    ALLOCM          ; ALLOCATE THE MEMORY
 004502  104070  004702                        MOV     R0,SECPTR       ; SECPTR POINTS TO WRITE BUFFER
 004504  104207  000007          21$:          MOV     #LINKLN,R0      ; R0 HAS LENGTH OF CHAIN LINK
 004506  024674                                CALL    ALLOCM          ; ALLOCATE THE MEMORY
 004507  104070  002232                        MOV     R0,CHAINS       ; CHAINS POINTS TO FIRST LINK
 004511  004532                                BR      24$             ; BRANCH
 004512  104171                  22$:          MOV     (R0),R1         ; R1 HAS NEXT LINK POINTER AND STATUS
 004513                                        ASSUME  RW.STAT,0       ; ASSUME STATUS AND POINTER FIRST WORD
 004513  102201  100000                        BIT     #EOC,R1         ; SEE IF THIS LINK IS THE LAST
 004515  054522                                BNE     23$             ; IF SO, BRANCH
 004516  103201  170000                        BIC     #UNADDR,R1      ; CLEAR UNUSED ADDRESSING BITS
 004520  104017                                MOV     R1,R0           ; R0 POINTS TO NEXT LINK
 004521  004512                                BR      22$             ; LOOP
 004522  104072                  23$:          MOV     R0,R2           ; SAVE R0
 004523  104207  000007                        MOV     #LINKLN,R0      ; R0 HAS LENGTH OF LINK
 004525  024674                                CALL    ALLOCM          ; ALLOCATE MEMORY FOR LINK
 004526  103201  100000                        BIC     #EOC,R1         ; CLEAR THE END-OF-CHAIN FLAG
 004530  101071                                BIS     R0,R1           ; PUT IN POINTER TO NEXT LINK
 004531  100121                                MOV     R1,(R2)         ; PUT POINTER BACK IN LAST LINK
 004532                                        ASSUME  RW.STAT,0       ; ASSUME STATUS AND POINTER FIRST WORD
 004532                          24$:
```

```
                                .SBTTL  FILLIN - FILL THE READ/WRITE CHAIN LINK (NODE) WITH REQUIRED INFORMATION
                                ;FILLIN
                                ;
                                ;       FILLIN BUILDS THE PARAMETERS THE UDA REQUIRES TO WRITE OR READ A BLOCK
                                ;
 6 004532  104302  002204              MOV     TRACK,R2        ; GET TRACK (HEAD) NUMBER
 7 004534  102203  000100              BIT     #REDWRT,R3      ; SEE IF READ IN PROGRESS
 8 004536  014553                      BEQ     11$             ; IF SO, BRANCH
 9 004537  102203  000400              BIT     #REVEC,R3       ; SEE IF TRYING TO FIND A REPLACEMENT
10 004541  054553                      BNE     11$             ; IF SO, YOU'RE TRYING TO READ, SO BRANCH
11 004542  101202  122400              BIS     #WREAL,R2       ; BUILD WRITE REAL TIME COMMAND
12 004544  100672  000004              MOV     R2,RW.CMD(R0)   ; MOVE TO SDI REAL TIME COMMAND AREA
13 004546  104202  140000              MOV     #WSTOP,R2       ; MOVE LAST BLOCK FLAG TO R2
14 004550  104301  004702              MOV     SECPTR,R1       ; R1 POINTS TO WRITE BUFFER
15 004552  004567                      BR      12$             ; BRANCH
16 004553  101202  013400  11$:       BIS     #RREAL,R2       ; BUILD READ REAL TIME COMMAND
17 004555  100672  000004              MOV     R2,RW.CMD(R0)   ; MOVE TO SDI REAL TIME COMMAND AREA
18 004557  104202  100000              MOV     #RSTOP,R2       ; MOVE LAST BLOCK FLAG TO R2
19 004561                              PUSH    R0              ; SAVE R0
   004561  100467                                                                  MOV R0,-(SP)
20 004562  104207  000415              MOV     #RBUFLN,R0      ; SIZE OF READ BUFFER
21 004564  024674                      CALL    ALLOCM          ; ALLOCATE BUFFER
22 004565  104071                      MOV     R0,R1           ; R1 POINTS TO BUFFER
23 004566                              POP     R0              ; RESTORE R0
   004566  104267                                                                  MOV (SP)+,R0
24 004567  100172  12$:       MOV     R2,(R0)         ; MOVE LAST BLOCK FLAG TO NEXT BLOCK POINTER
28 004570                              ASSUME  RW.STAT,0
37 004570  100671  000001              MOV     R1,RW.BUF(R0)   ; MOVE POINTER TO SECTOR TO POINTER AREA
38 004572  104202  002207              MOV     #DUMSDI,R2      ; R2 POINTS TO DUMMY SDI AREA
39 004574  100672  000005              MOV     R2,RW.SDI(R0)   ; MOVE R2 TO POINTER TO DUMMY SDI AREA
40 004576  104302  002205              MOV     SECTOR,R2       ; R2 CONTAINS SECTOR TO BE WRITTEN
41 004600  100672  000006              MOV     R2,RW.ANG(R0)   ; MOVE TO ANGLE FROM INDEX
42 004602  104302  002175              MOV     CURBN,R2        ; MOVE LOW ORDER BN TO R2
43 004604  100672  000002              MOV     R2,RW.LOW(R0)   ; MOVE LOW ORDER BN TO LOW EXPECTED HEADER
44 004606  104302  002176              MOV     CURBN+1,R2      ; MOVE HIGH ORDER BN TO R2
45 004610  102203  000200              BIT     #RBNBN,R3       ; SEE IF BN IS AN RBN
46 004612  054622                      BNE     15$             ; IF SO, BRANCH
47 004613  104143                      MOV     (R4),R3         ; GET SUBUNIT PARAMETERS
48 004614                              ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
49 004614  102203  020000              BIT     #DCYLS,R3       ; SEE IF USING DIAGNOSTIC CYLINDERS
50 004616  014624                      BEQ     16$             ; IF NOT, BRANCH
51 004617  101202  140000              BIS     #HD.DBN,R2      ; DIAGNOSTIC HEADER
52 004621  004624                      BR      16$             ; BRANCH
53 004622  101202  060000  15$:       BIS     #HD.RBN,R2      ; REVECTORED HEADER
54 004624  105302  002172  16$:       ADD     HIBN,R2         ; ADD HI BN BITS
55 004626                              ASSUME  HD.LBN,0
56 004626  100672  000003              MOV     R2,RW.HI(R0)    ; MOVE R2 TO HI WORD EXPECTED HEADER
57 004630  104302  002206              MOV     INDEX,R2        ; GET DISTANCE FROM INDEX
58 004632  115402                      INC     R2              ; ADJUST INDEX
59 004633  106302  002217              CMP     SCR1,R2         ; SEE IF FIRST SECTOR AFTER INDEX
60 004635  054637                      BNE     13$             ; IF NOT, BRANCH
61 004636  114002                      CLR     R2              ; ZERO THETA FROM INDEX
62 004637  100672  000006  13$:       MOV     R2,RW.ANG(R0)   ; SAVE
71 004641  107200  000001  002235      SUB     #1,MAXSEC       ; DECREMENT SECTOR COUNT
72 004644  014654                      BEQ     7$              ; IF COUNT COMPLETE, BRANCH
73 004645  105200  000001  002175      ADD     #1,CURBN        ; ADD ONE TO LO CURRENT SECTOR
74 004650  044447                      BCC     5$              ; IF NO CARRY, BRANCH
```

```
     75 004651  115400  002176                 INC     CURBN+1          ; PROPOGATE CARRY
     76 004653  004447                         BR      5$               ; BRANCH
     77 004654  114001                  7$:    CLR     R1               ; IMMIDATE CALL
     78 004655  114002                         CLR     R2               ; NO ERRORS
     79 004656  104653  000046                 MOV     U.PARM(R5),R3    ; GET UNIT PARAMETERS
     80 004660  102203  000100                 BIT     #REDWRT,R3       ; SEE IF READ IS TO BE DONE
     81 004662  014671                         BEQ     2$               ; IF SO, BRANCH
     82 004663  102203  000400                 BIT     #REVEC,R3        ; SEE IF READING RCT
     83 004665  054671                         BNE     2$               ; IF SO, BRANCH
     84 004666  104207  000010                 MOV     #WRITE,R0        ; WRITE ROUTINE IS NEXT
     85 004670  004673                         BR      3$               ; BRANCH
     86 004671  104207  000012          2$:    MOV     #READ,R0         ; READ ROUTINE IS NEXT
     87 004673  003231                  3$:    BR      JMPRET
     88                                        .DSABL  LSB
```

```
                                        .SBTTL  ALLOCM - ALLOCATE MEMORY FOR THE READ/WRITE BUFFERS AND CHAIN
004674                                  ALLOCM:
                                        ;
                                        ;       ALLOCATE MEMORY FOR READ/WRITE BUFFERS
                                        ;
004674  107070  004701                          SUB     R0,TMEMPL       ; SUBTRACT LENGTH FROM MEMORY POOL
004676  104307  004701                          MOV     TMEMPL,R0       ; R0 POINTS TO MEMORY ALLOCATED
004700  000000                                RETURN                  ; RETURN TO CALLING MODULE

004701                                  TMEMPL: .BLKW   1                       ; TEMPORARY MEMORY POOL FOR READ/WRITE
004702                                  SECPTR: .BLKW   1                       ; SECTOR POINTER FOR WRITE
                                                .IF     LT,BUFARA-.
                                        BUFARA  =       .
                                                .ENDC
                                                .IF     LE,MAXADR-.
                                        MAXADR  =       .+1
                                                .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE WRITE
                                        DMOVLY  W,AREA0
004703  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
000010                          WRITE   =       BUILDP+1
                                ;
                                ;       WRITE GENERATES THE PATTERNS AND WRITES THEM TO THE DRIVE
                                ;
                                        .ENABL  LSB
004413  024172                          CALL    DSABLE          ; DISABLE ERROR RECOVERY
004414  104657  000012                  MOV     U.RWTO(R5),R0   ; MOVE READ/WRITE TIMEOUT VALUE TO R0
004416  115407                          INC     R0              ; INCREMENT COUNT
004417  100657  000012                  MOV     R0,U.RWTO(R5)   ; SAVE
004421  014513                          BEQ     8$              ; IF ZERO, WE KNOW THAT THE TIMEOUT IS UNEX
004422  106207  000002                  CMP     #2,R0           ; SEE IF TRIED MAXIMUM TIMES
004424                                  BCS     11$             ; IF SO, BRANCH
004424  044426                                                                  BCC     .+2
004425  004432                                                                  BR      11$
004426  104141                          MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
004427                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
004427  102201  001000                  BIT     #RTRIES,R1      ; SEE IF RETRIES ARE ENABLED
004431  054513                          BNE     8$              ; IF SO, BRANCH
004432                          11$:    HARDER  25,<S.LETR(R4),U.CBN(R5),U.CBN+1(R5)>
004432  104200  002337  001107                                                  MOV     #ER25,HRQ.04
004435  104640  000005  001110                                                  MOV     S.LETR(R4),HRQ.05
004440  104650  000053  001111                                                  MOV     U.CBN(R5),HRQ.06
004443  104650  000054  001112                                                  MOV     U.CBN+1(R5),HRQ.07
004446  104202  107671                                                          MOV     #25!ERHARD+4000.,R2
004450  104020  001105                                                          MOV     R2,HRQ.02
004452  104200  004452  001104                                                  MOV     #.,HRQ.01
004455  104200  060014  001103                                                  MOV     #ERRMC,HRQ.RQ
004460                                  ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5),U.RBN+1(R5)>
004460  104650  000046  001113                                                  MOV     U.PARM(R5),HRQ.08
004463  104200  005472  001114                                                  MOV     #RBNTXT,HRQ.09
004466  104650  000055  001115                                                  MOV     U.RBN(R5),HRQ.10
004471  104650  000056  001116                                                  MOV     U.RBN+1(R5),HRQ.11
004474                                  ENDERR  0
004474  114000  002230                                                          CLR     ERRPOS          ; CLEAR THE POSITION
004476  104653  000046                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
004500  103203  000200                  BIC     #RBNBN,R3       ; IF HANDLING AN RBN, CLEAR IT
004502  100653  000046                  MOV     R3,U.PARM(R5)   ; SAVE
004504  104201  000001                  MOV     #1,R1           ; ONLY 1 SECTOR HANDLED
004506  100651  000021                  MOV     R1,U.NSEC(R5)   ; STORE
004510  104207  000001                  MOV     #SETUP,R0       ; SETUP IS NEXT MODULE CALLED
004512  004636                          BR      7$              ; BRANCH
004513  021777                  8$:     CALL    BULDUM          ; BUILD DUMMY SDI CONTROL BLOCK

                                .SBTTL  BULDSC - BUILD THE SECTOR TO WRITE (FILL WITH PATTERN AND CALC EDC)
                                ;BULDSC
                                ;
                                ;       BULDSC BUILDS THE INFORMATION AND DATA PATTERN TO BE WRITTEN ON THE
```

```
                                        ;       SECTOR
                                        ;
004514  104651  000014                        MOV     U.PAT(R5),R1    ; GET PATTERN NUMBER TO WRITE
004516  104013                                MOV     R1,R3           ; COPY PATTERN NUMBER TO R3
004517  110701                                SWAB    R1              ; SWAP THE BYTES
004520  101013                                BIS     R1,R3           ; REPLICATE THE PATTERN NUMBER
004521  104031                                MOV     R3,R1           ; COPY TO R1
004522  110201                                ROL     R1              ; ROTATE TO NEXT POSITION
004523  110201                                ROL     R1
004524  110201                                ROL     R1
004525  110201                                ROL     R1
004526  103201  007417                        BIC     #^CHBLONB!^CLBLONB,R1   ; CLEAR UNUSED BITS
004530  101013                                BIS     R1,R3           ; REPLICATE THE PATTERN
004531  104307  002232                        MOV     CHAINS,R0       ; GET POINTER TO FIRST WRITE CHAIN NODE
004533  104677  000001                        MOV     RW.BUF(R0),R0   ; R0 POINTS TO BUFFER AREA
004535  100273                                MOV     R3,(R0)+        ; MOVE PATTERN NUMBERS TO SECTOR
                                        .SBTTL  COPPAT - COPY THE DATA PATTERN TO BUFFER
                                        ;COPPAT
                                        ;
                                        ;       COPPAT COPIES THE PATTERN TO THE SECTOR TO BE WRITTEN
                                        ;
004536                                  PUSH    <R0,R4,R5>      ; SAVE R0,R4,R5
004536  100467                                                                  MOV R0,-(SP)
004537  100464                                                                  MOV R4,-(SP)
004540  100465                                                                  MOV R5,-(SP)
004541  104652  000014                        MOV     U.PAT(R5),R2    ; R2 HAS PATTERN NUMBER
004543  104622  002236                        MOV     PATPTR(R2),R2   ; POINT TO PATTERN
004545  115402                                INC     R2              ; SKIP EDC
004546  104201  000377                        MOV     #SCTWRD,R1      ; R1 HAS NUMBER OF WORDS TO FILL WITH PATTERN
004550  104024                        COPLP0: MOV     R2,R4           ; R4 POINTS TO LENGTH OF PATTERN
004551  104243                                MOV     (R4)+,R3        ; R3 CONTAINS LENGTH OF PATTERN
004552  106203  000001                        CMP     #1,R3           ; SEE IF PATTERN IS 1 WORD LONG
004554  014564                                BEQ     ONEPAT          ; IF SO, BRANCH
004555  104245                        COPLP1: MOV     (R4)+,R5        ; R5 GETS 1 WORD OF THE DATA PATTERN
004556  100275                                MOV     R5,(R0)+        ; MOVE PATTERN WORD TO SECTOR AREA
004557  117401                                DEC     R1              ; DECREMENT NUMBER OF WORDS TO WRITE IN SECTOR
004560  014570                                BEQ     COPFIN          ; IF ALL WORDS WRITTEN, BRANCH
004561  117403                                DEC     R3              ; DECREMENT COUNT OF WORDS IN PATTERN
004562  054555                                BNE     COPLP1          ; IF DATA PATTERN UNFINISHED, BRANCH
004563  004550                                BR      COPLP0          ; BRANCH
004564  104145                        ONEPAT: MOV     (R4),R5         ; GET 1 WORD OF DATA PATTERN
004565  100275                        COPLP2: MOV     R5,(R0)+        ; MOVE PATTERN TO SECTOR AREA
004566  117401                                DEC     R1              ; DECREMENT NUMBER OF WORDS TO MOVE TO SECTOR
004567  054565                                BNE     COPLP2          ; IF INCOMPLETE, BRANCH
004570  104425                        COPFIN: MOV     -(R2),R5        ; GET EDC
004571  100175                                MOV     R5,(R0)         ; MOVE TO BUFFER
004572                                  POP     <R5,R4,R0>      ; RESTORE R5,R4,R0
004572  104265                                                                  MOV (SP)+,R5
004573  104264                                                                  MOV (SP)+,R4
004574  104267                                                                  MOV (SP)+,R0
004575  104651  000014                        MOV     U.PAT(R5),R1    ; GET PATTERN NUMBER
004577  054604                                BNE     2$              ; IF NOT PATTERN 16, BRANCH
004600  117407                                DEC     R0              ; POINT TO START OF BUFFER
004601  021146                                CALL    CMPEDC          ; COMPUTE EDC
004602  100672  000400                        MOV     R2,BF.EDC(R0)   ; SAVE EDC
004604  024637                        21$:    CALL    WBLOCK          ; WRITE THE SECTOR
004605  115002                                TST     R2              ; SEE IF ERROR OCCURRED
```

```
117 004606  054636                          BNE     7$              ; IF NOT, BRANCH
118 004607  104643  000011                  MOV     S.MEGW(R4),R3   ; GET MEGABYTE COUNT
119 004611  106203  003642                  CMP     #1954.,R3       ; SEE IF ONE MEGABYTE TRANSFERED
120 004613  034632                          BPL     6$              ; BRANCH IF NOT
121 004614  107203  003642                  SUB     #1954.,R3       ; CLEAR MEGABYTE COUNT
122 004616  100643  000011                  MOV     R3,S.MEGW(R4)   ; SAVE NEW COUNT
123 004620  114000  001105                  CLR     HRQ.02          ; NOT REPORTING READS
124 004622  104200  000001  001106          MOV     #1,HRQ.03       ; REPORT 1 MEGABIT WRITTEN
125 004625  104202  060011                  MOV     #T4MXFR,R2      ; SET UP FOR MEGABIT REPORT
126 004627  104020  001103                  MOV     R2,HRQ.RQ       ; FLAG AS NON-ERROR
127 004631  004633                          BR      1$              ; EXIT
128 004632  114002                  6$:     CLR     R2              ; NO ERRORS
129 004633  104207  000011          1$:     MOV     #AFTWRT,R0      ; AFTWRT IS NEXT MODULE
130 004635  114001                  2$:     CLR     R1              ; IMMIDATE CALL
131 004636  003231                  7$:     BR      JMPRET          ; RETURN TO RDWRT
132                                         .DSABL  LSB
```

```
                                .SBTTL  WBLOCK - WRITE THE SECTOR(S)
004637                          WBLOCK:
                                ;
                                ;       WBLOCK WRITES THE SECTOR(S) ONTO THE DRIVE
                                ;
004637                                  PUSH    R4              ; SAVE R4
004637  010464                                                          MOV R4,-(SP)
004640  104307  002232                  MOV     CHAINS,R0       ; POINT TO START OF CHAIN
004642  104652  000025                  MOV     U.MASK(R5),R2   ; R2 HAS SDI INTERCONNECT
004644  104644  000007                  MOV     S.SCHR(R4),R4   ; R4 POINTS TO SUBUNIT CHARACTERISTICS
004646  104644  000005                  MOV     DATPRE(R4),R4   ; R4 CONTAINS DATA PREAMBLE LENGTH
004650  103204  177400                  BIC     #HIBYTE,R4      ; STRIP OFF UNUSED BITS
004652  060012                          XFC     WAITSI          ; WAIT FOR SECTOR OR INDEX PULSE
004653  115001                          TST     R1              ; SEE IF ERROR OCCURRED
004654  014714                          BEQ     3$
004655                                  DEVFTL  42,<U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
004655  104200  003066  001107                                          MOV     #ER42,HRQ.04
004660  104650  000066  001110                                          MOV     U.CGRP(R5),HRQ.05
004663  104650  000064  001111                                          MOV     U.CCYL(R5),HRQ.06
004666  104650  000065  001112                                          MOV     U.CCYL+1(R5),HRQ.07
004671  104202  047712                                                  MOV     #42!FTLDEV+4000.,R2
004673  104020  001105                                                  MOV     R2,HRQ.02
004675  104200  004675  001104                                          MOV     #.,HRQ.01
004700  104200  060014  001103                                          MOV     #ERRMC,HRQ.RQ
004703                                  ENDERR
004703  104200  000051  001113                                          MOV     #SER22,HRQ.08
004706  104200  000011  002230                                          MOV     #8+1,ERRPOS     ; SET THE POSITION
004711  024207                          CALL    GOSEEK
004712                                  POP     R4              ; RESTORE SUBUNIT POINTER
004712  104264                                                          MOV (SP)+,R4
004713  005015                          BR      6$              ; EXIT
004714                          3$:
004714  060003                          XFC     XWRITE          ; WRITE THE SECTOR(S)
004715                                  POP     R4              ; RESTORE R4
004715  104264                                                          MOV (SP)+,R4
004716  100651  000061                  MOV     R1,U.RWER(R5)   ; SAVE ERROR IF ANY
004720  014767                          BEQ     5$              ; IF NO ERROR, BRANCH
004721  104651  000026                  MOV     U.WRIT(R5),R1   ; GET WRITE PROTECTION STATUS
                                        ;       NOW I OR IN THE READ-ONLY WRITE PROTECT BITS.  THIS WILL CATCH
                                        ;       IT IF SOMEHOW I GET INTO THIS MODULE ON A READ-ONLY DRIVE
                                        ;       (I HOPE NOT)
004723  101651  000045                  BIS     U.WPRT(R5),R1   ; SET READ-ONLY BITS
004725  104652  000050                  MOV     U.SUBU(R5),R2   ; GET SUBUNIT IN USE
004727  105202  000003                  ADD     #3,R2           ; ADD 3 TO SHIFT WRITE PROT STAT TO LO NIBBLE
004731  110601                  10$:    ROR     R1              ; ROTATE PROTECTION STATUS (LO BIT TO CARRY)
004732  117402                          DEC     R2              ; DECREMENT SUBUNIT IN USE
004733  034731                          BPL     10$             ; IF >= 0, BRANCH
004734  110601                          ROR     R1              ; MOVE THE BIT INTO CARRY TO TEST
004735  045014                          BCC     7$              ; IF CARRY CLEAR (NOT WRITE PROTECTED) BRANCH
004736                                  DEVFTL  18,U.RWER(R5)   ; REPORT WRITE ON WRITE PROTECTED DRIVE
004736  104200  001702  001107                                          MOV     #ER18,HRQ.04
004741  104650  000061  001110                                          MOV     U.RWER(R5),HRQ.05
004744  104202  047662                                                  MOV     #18!FTLDEV+4000.,R2
004746  104020  001105                                                  MOV     R2,HRQ.02
004750  104200  004750  001104                                          MOV     #.,HRQ.01
004753  104200  060014  001103                                          MOV     #ERRMC,HRQ.RQ
004756                                  ENDERR
```

WBLOCK - WRITE THE SECTOR(S)

```
                004756  104200  000051  001111                                          MOV     #SER22,HRQ.06
                004761  104200  000007  002230                                          MOV     #6+1,ERRPOS     ; SET THE POSITION
     46         004764  024207                          CALL    GOSEEK
     47         004765  104051                          MOV     R5,R1           ; DEFFERED CALL
     48         004766  005015                          BR      6$              ; BRANCH
     49         004767  104657  000012          5$:     MOV     U.RWTO(R5),R0   ; GET TIMEOUTS
     50         004771  015014                          BEQ     7$              ; IF NO RETRIES, BRANCH
     51         004772                          REPSFT  SOFT            ; REPORT ONE SOFT ERROR
                004772  104200  000001  001105                                          MOV     #1,HRQ.02
                004775  114000  001106                                                  CLR     HRQ.03
                004777  114000  001107                                                  CLR     HRQ.04
                005001  100467                                                                  MOV R0,-(SP)
                005002  104657  000063                                                  MOV     U.UNUM(R5),R0
                005004  105657  000050                                                  ADD     U.SUBU(R5),R0
                005006  104070  001104                                                  MOV     R0,HRQ.01
                005010  104207  060007                                                  MOV     #T4SOFT,R0
                005012  021053                                                          CALL    HOSTRQ
                005013  104267                                                                  MOV (SP)+,R0
     52         005014  114002          7$:     CLR     R2              ; NO ERRORS
     53         005015  000000          6$:     RETURN                  ; RETURN TO CALLING PROGRAM;
     57                                          .IF     LT,BUFARA-.
     58                                  BUFARA  =       .
     59                                          .ENDC
     60                                          .IF     LE,MAXADR-.
     61                                  MAXADR  =       .+1
     62                                          .ENDC
```

```
********** OVERLAY MODULE AFTWRT

                                .SBTTL  ********** OVERLAY MODULE AFTWRT
                                        DMOVLY  AW,AREA0
005016  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000011                          AFTWRT  =       WRITE+1
                                ;
                                ;       AFTWRT WILL HANDLE ANY ERRORS THAT OCCURRED DURING THE WRITE
                                ;       OPERATION
                                ;
                                        .ENABL  LSB
                                .SBTTL  FNDWER - IF ERROR DURING WRITE, FIND IT'S POSITION IN THE CHAIN
                                ;FNDWER
                                ;
                                ;       FIND THE FIRST BUFFER THAT IS NOT WRITTEN
                                ;
004413                          PUSH    <R4,R5>         ; SAVE REGISTERS
004413  100464                                                          MOV R4,-(SP)
004414  100465                                                          MOV R5,-(SP)
004415  104654  000053                 MOV     U.CBN(R5),R4    ; GET LO ORDER BN
004417  104655  000054                 MOV     U.CBN+1(R5),R5  ; GET HI BN
004421  114002                         CLR     R2              ; CLEAR SECTOR COUNT
004422  104307  002232                 MOV     CHAINS,R0       ; R0 POINTS TO FIRST BUFFER
004424  100674  000002         31$:    MOV     R4,RW.LOW(R0)   ; MOVE TO CHAIN
004426  100675  000003                 MOV     R5,RW.HI(R0)    ; MOVE TO CHAIN
004430  104171                         MOV     (R0),R1         ; GET STATUS BITS
004431                                 ASSUME  RW.STAT,0
004431  102201  040000                 BIT     #BUFFLG,R1      ; SEE IF THIS BUFFER HAS BEEN WRITTEN
004433  054447                         BNE     32$             ; IF NOT, BRANCH
004434  115402                         INC     R2              ; INCREMENT SECTOR COUNT
004435  115001                         TST     R1              ; SEE IF END-OF-LIST
004436  074447                         BMI     32$             ; IF LAST BUFFER, EXIT
004437                                 ASSUME  EOC,100000      ; ASSUME WRITE STOP BIT IS SIGN BIT
004437  103201  170000                 BIC     #UNADDR,R1      ; CLEAR UNUSED ADDRESSING BITS
004441  104017                         MOV     R1,R0           ; MOVE TO R0
004442  105204  000001                 ADD     #1,R4           ; ADD ONE TO LOW ORDER BN
004444  044424                         BCC     31$             ; IF NO CARRY, BRANCH
004445  115405                         INC     R5              ; PRPOGATE CARRY
004446  004424                         BR      31$             ; LOOP
004447                          32$:    POP     <R5,R4>         ; RESTORE REGISTERS
004447  104265                                                          MOV (SP)+,R5
004450  104264                                                          MOV (SP)+,R4
004451  100652  000021                 MOV     R2,U.NSEC(R5)   ; SAVE AS NUMBER OF SECTORS HANDLED
004453  105642  000011                 ADD     S.MEGW(R4),R2   ; ADD TO MEGABITS WRITTEN
004455  100642  000011                 MOV     R2,S.MEGW(R4)   ; SAVE
004457  104652  000061                 MOV     U.RWER(R5),R2   ; SEE IF ANY ERROR OCCURRED
004461  054474                         BNE     3$              ; IF SO, BRAN.H
004462  104657  000046                 MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
004464  103207  000200                 BIC     #RBNBN,R0       ; CLEAR RBN BIT
004466  100657  000046                 MOV     R0,U.PARM(R5)   ; SAVE
004470  104207  000001                 MOV     #SETUP,R0       ; SETUP NEXT MODULE
004472  104051                         MOV     R5,R1           ; DEFERRED CALL
```

```
     88 004473  005214                        BR      17$             ; EXIT
     89 004474  021725                3$:     CALL    BLKCHK          ; SEE IF THIS IS A KNOWN BAD BLOCK
     90 004475                               BCS     15$             ; IF NOT, BRANCH
        004475  044477                                                                  BCC     .+2
        004476  004510                                                                  BR      15$
     91 004477  104657  000021               MOV     U.NSEC(R5),R0   ; GET NUMBER OF SECTORS WRITTEN
     92 004501  115407                       INC     R0              ; SKIP THIS SECTOR
     93 004502  100657  000021               MOV     R0,U.NSEC(R5)   ; SAVE
     94 004504  104207  000001               MOV     #SETUP,R0       ; SETUP NEXT MODULE
     95 004506  104051                       MOV     R5,R1           ; DEFERRED CALL
     96 004507  005214                       BR      17$             ; EXIT
     97 004510  104651  000021        15$:   MOV     U.NSEC(R5),R1   ; GET SECTORS WRITTEN
     98 004512  105651  000024               ADD     U.CSEC(R5),R1   ; ADD TO TOTAL SECTORS WRITTEN
     99 004514  100651  000024               MOV     R1,U.CSEC(R5)   ; SAVE
    100 004516  104651  000021               MOV     U.NSEC(R5),R1   ; GET SECTORS WRITTEN
    101 004520  105651  000053               ADD     U.CBN(R5),R1    ; ADD TO LO ORDER CURRENT BN
    102 004522  100651  000053               MOV     R1,U.CBN(R5)    ; SAVE
    103 004524  044532                       BCC     2$              ; IF NO CARRY, BRANCH
    104 004525  104651  000054               MOV     U.CBN+1(R5),R1  ; GET HI ORDER CURRENT BN
    105 004527  115401                       INC     R1              ; PROPOGATE CARRY
    106 004530  100651  000054               MOV     R1,U.CBN+1(R5)  ; SAVE
    107 004532  114001                2$:    CLR     R1              ; SET UP FOR NO SECTORS WTRITTEN
    108 004533  100651  000021               MOV     R1,U.NSEC(R5)   ; SAVE
    109 004535  104651  000061               MOV     U.RWER(R5),R1   ; GET WRITE ERROR
    110 004537  106201  000003               CMP     #3,R1           ; SEE IF REVECTOR
    111 004541  014600                       BEQ     1$              ; IF SO, BRANCH
    112 004542  106201  000002               CMP     #2,R1           ; SEE IF FAILURE IN DRIVE
    113 004544  054564                       BNE     6$              ; IF NOT, BRANCH
    114 004545                               SOFTER  6               ; REPORT
        004545  104200  000562  001107                                          MOV     #ER6,HRQ.04
        004550  104202  147646                                                   MOV     #6!ERSOFT+4000.,R2
        004552  104020  001105                                                   MOV     R2,HRQ.02
        004554  104200  004554  001104                                          MOV     #.,HRQ.01
        004557  104200  060013  001103                                          MOV     #ERRMES,HRQ.RQ
    115 004562  024223                       CALL    GOINIT
    116 004563  005115                       BR      12$             ; BRANCH
    117 004564  106201  000004        6$:    CMP     #4,R1           ; SEE IF HEADER COMPARE FAILURE
    118 004566  054712                       BNE     8$              ; IF NOT, BRANCH
    119 004567  104651  000046               MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
    120 004571  104143                       MOV     (R4),R3         ; GET SUBUNIT PARAMETERS
    121 004572                               ASSUME  S.PARM,0
    122 004572  102203  020000               BIT     #DCYLS,R3       ; SEE IF USING DIAGNOSTIC CYLINDERS
    123 004574  054604                       BNE     7$              ; IF SO, BRANCH
    124 004575  102201  000200               BIT     #RBNBN,R1       ; SEE IF WRITING A RBN
    125 004577  054604                       BNE     7$              ; IF SO, BRANCH
    126 004600  104207  000022        1$:    MOV     #REVCT,R0       ; REVECTOR NEXT MODULE
    127 004602  114001                       CLR     R1              ; IMMIDATE CALL
    128 004603  005214                       BR      17$             ; EXIT
    129 004604                        7$:    HARDER  5               ; REPORT HEADER COMPARE FAILURE
        004604  104200  000365  001107                                          MOV     #ER5,HRQ.04
        004607  104202  107645                                                   MOV     #5!ERHARD+4000.,R2
        004611  104020  001105                                                   MOV     R2,HRQ.02
        004613  104200  004613  001104                                          MOV     #.,HRQ.01
        004616  104200  060014  001103                                          MOV     #ERRMC,HRQ.RQ
    130 004621                               ERRORC  <S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
        004621  104640  000005  001110                                          MOV     S.LETR(R4),HRQ.05
        004624  104670  000002  001111                                          MOV     RW.LOW(R0),HRQ.06
```

```
     004627  104670  000003  001112                          MOV     RW.HI(R0),HRQ.07
131  004632                          ERRORC  <R1,#RBNTXT,U.RBN(R5)>
     004632  104010  001113                                  MOV     R1,HRQ.08
     004634  104200  005472  001114                          MOV     #RBNTXT,HRQ.09
     004637  104650  000055  001115                          MOV     U.RBN(R5),HRQ.10
132  004642                          ERRORC  <U.RBN+1(R5),RW.ANG(R0)>
     004642  104650  000056  001116                          MOV     U.RBN+1(R5),HRQ.11
     004645  104670  000006  001117                          MOV     RW.ANG(R0),HRQ.12
133  004650                          ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
     004650  104670  000004  001120                          MOV     RW.CMD(R0),HRQ.13
     004653  104650  000066  001121                          MOV     U.CGRP(R5),HRQ.14
     004656  104650  000064  001122                          MOV     U.CCYL(R5),HRQ.15
     004661  104650  000065  001123                          MOV     U.CCYL+1(R5),HRQ.16
134  004664                          ERRORC  <U.LGRP(R5),U.LCYL(R5),U.LCYL+1(R5)>
     004664  104650  000071  001124                          MOV     U.LGRP(R5),HRQ.17
     004667  104650  000067  001125                          MOV     U.LCYL(R5),HRQ.18
     004672  104650  000070  001126                          MOV     U.LCYL+1(R5),HRQ.19
135  004675                          ENDERR  0
     004675  114000  002230                                  CLR     ERRPOS          ; CLEAR THE POSITION
136  004677  103201  000200          BIC     #RBNBN,R1       ; NO LONGER HANDLING AN RBN
137  004701  100651  000046          MOV     R1,U.PARM(R5)   ; SAVE
138  004703  104201  000001          MOV     #1,R1           ; ONLY 1 SECTOR WRITTEN
139  004705  100651  000021          MOV     R1,U.NSEC(R5)   ; SAVE
140  004707  104207  000001          MOV     #SETUP,R0       ; SETUP NEXT MODULE
141  004711  005215                  BR      13$             ; EXIT AND REPORT
142  004712  106201  000153    8$:   CMP     #153,R1         ; SEE IF POSITIONER ERROR
143  004714  054756                  BNE     22$             ; IF NOT, BRANCH
144  004715                          REPSFT  ,,SEEK          ; REPORT SEEK ERROR
     004715  114000  001105                                  CLR     HRQ.02
     004717  114000  001106                                  CLR     HRQ.03
     004721  104200  000001  001107                          MOV     #1,HRQ.04
     004724  100467                                                  MOV R0,-(SP)
     004725  104657  000063                                  MOV     U.UNUM(R5),R0
     004727  105657  000050                                  ADD     U.SUBU(R5),R0
     004731  104070  001104                                  MOV     R0,HRQ.01
     004733  104207  060007                                  MOV     #T4SOFT,R0
     004735  021053                                          CALL    HOSTRQ
     004736  104267                                                  MOV (SP)+,R0
145  004737                          SOFTER  44              ; POSITIONER ERROR
     004737  104200  003127  001107                          MOV     #ER44,HRQ.04
     004742  104202  147714                                  MOV     #44!ERSOFT+4000.,R2
     004744  104020  001105                                  MOV     R2,HRQ.02
     004746  104200  004746  001104                          MOV     #.,HRQ.01
     004751  104200  060013  001103                          MOV     #ERRMES,HRQ.RQ
146  004754  024215                  CALL    GORCLB          ; RECALIBRATE NEEDED
147  004755  005115                  BR      12$             ; BRANCH
148  004756  106201  000213    22$:  CMP     #213,R1         ; SEE IF READ/WRITE FAILURE
149  004760  054777                  BNE     23$             ; IF NOT, BRANCH
150  004761                          SOFTER  48              ; READ/WRITE READY FAILURE
     004761  104200  003243  001107                          MOV     #ER48,HRQ.04
     004764  104202  147720                                  MOV     #48!ERSOFT+4000.,R2
     004766  104020  001105                                  MOV     R2,HRQ.02
     004770  104200  004770  001104                          MOV     #.,HRQ.01
     004773  104200  060013  001103                          MOV     #ERRMES,HRQ.RQ
151  004776  005115                  BR      12$             ; ERROR EXIT
152  004777  106201  000253    23$:  CMP     #253,R1         ; SEE IF DRIVE DATA OR CLOCK TIMEOUT
153  005001  055021                  BNE     21$             ; IF NOT, BRANCH
```

```
154 005002                                      SOFTER  47              ; DRIVE CLOCK TIMEOUT
    005002  104200  003214  001107                                              MOV     #ER47,HRQ.04
    005005  104202  147717                                               MOV     #47!ERSOFT+4000.,R2
    005007  104020  001105                                               MOV     R2,HRQ.02
    005011  104200  005011  001104                                       MOV     #.,HRQ.01
    005014  104200  060013  001103                                       MOV     #ERRMES,HRQ.RQ
155 005017  024223                              CALL    GOINIT          ; INIT THE DRIVE TO RECOVER
156 005020  005115                              BR      12$             ; ERROR EXIT
157 005021  106201  000313          21$:        CMP     #313,R1         ; SEE IF RECIEVER READY TIMEOUT
158 005023  055043                              BNE     24$             ; IF NOT, BRANCH
159 005024                                      SOFTER  49              ; RECEIVER READY FAILURE
    005024  104200  003270  001107                                              MOV     #ER49,HRQ.04
    005027  104202  147721                                               MOV     #49!ERSOFT+4000.,R2
    005031  104020  001105                                               MOV     R2,HRQ.02
    005033  104200  005033  001104                                       MOV     #.,HRQ.01
    005036  104200  060013  001103                                       MOV     #ERRMES,HRQ.RQ
160 005041  024223                              CALL    GOINIT          ; INIT THE DRIVE TO RECOVER
161 005042  005115                              BR      12$
162 005043  106201  000413          24$:        CMP     #413,R1         ; SEE IF RTDS ERROR DURING WRITE
163 005045  055064                              BNE     25$             ; IF NOT, BRANCH
164 005046                                      SOFTER  63              ; REPORT
    005046  104200  004115  001107                                              MOV     #ER63,HRQ.04
    005051  104202  147737                                               MOV     #63!ERSOFT+4000.,R2
    005053  104020  001105                                               MOV     R2,HRQ.02
    005055  104200  005055  001104                                       MOV     #.,HRQ.01
    005060  104200  060013  001103                                       MOV     #ERRMES,HRQ.RQ
165 005063  005115                              BR      12$             ; BRANCH
166 005064                          25$:        HARDER  68,<#SER36,R1>          ; UNKNOWN ERROR CODE
    005064  104200  004175  001107                                              MOV     #ER68,HRQ.04
    005067  104200  004242  001110                                       MOV     #SER36,HRQ.05
    005072  104010  001111                                               MOV     R1,HRQ.06
    005074  104202  107744                                               MOV     #68!ERHARD+4000.,R2
    005076  104020  001105                                               MOV     R2,HRQ.02
    005100  104200  005100  001104                                       MOV     #.,HRQ.01
    005103  104200  060014  001103                                       MOV     #ERRMC,HRQ.RQ
167 005106                                      ENDERR
    005106  104200  000051  001112                                       MOV     #SER22,HRQ.07
    005111  104200  000010  002230                                       MOV     #7+1,ERRPOS     ; SET THE POSITION
168 005114  005206                              BR      14$
169 005115                          12$:        CERROR  5,#SER19
    005115  104200  000470  001110                                       MOV     #SER19,HRQ.05
170 005120                                      ERRORC  <U.RWTO(R5),S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
    005120  104650  000012  001111                                       MOV     U.RWTO(R5),HRQ.06
    005123  104640  000005  001112                                       MOV     S.LETR(R4),HRQ.07
    005126  104670  000002  001113                                       MOV     RW.LOW(R0),HRQ.08
    005131  104670  000003  001114                                       MOV     RW.HI(R0),HRQ.09
171 005134                                      ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5)>
    005134  104650  000046  001115                                       MOV     U.PARM(R5),HRQ.10
    005137  104200  005472  001116                                       MOV     #RBNTXT,HRQ.11
    005142  104650  000055  001117                                       MOV     U.RBN(R5),HRQ.12
172 005145                                      ERRORC  <U.RBN+1(R5),RW.ANG(R0)>
    005145  104650  000056  001120                                       MOV     U.RBN+1(R5),HRQ.13
    005150  104670  000006  001121                                       MOV     RW.ANG(R0),HRQ.14
173 005153                                      ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
    005153  104670  000004  001122                                       MOV     RW.CMD(R0),HRQ.15
    005156  104650  000066  001123                                       MOV     U.CGRP(R5),HRQ.16
    005161  104650  000064  001124                                       MOV     U.CCYL(R5),HRQ.17
```

```
     005164  104650  000065  001125                                          MOV     U.CCYL+1(R5),HRQ.18
174  005167                          ERRORC  <U.LGRP(R5),U.LCYL(R5),U.LCYL+1(R5)>
     005167  104650  000071  001126                                          MOV     U.LGRP(R5),HRQ.19
     005172  104650  000067  001127                                          MOV     U.LCYL(R5),HRQ.20
     005175  104650  000070  001130                                          MOV     U.LCYL+1(R5),HRQ.21
175  005200                          ENDERR
     005200  104200  000051  001131                                          MOV     #SER22,HRQ.22
     005203  104200  000027  002230                                          MOV     #22+1,ERRPOS    ; SET THE POSITION
176  005206  024207                  14$:    CALL    GOSEEK          ; SEEK REQUIRED FOR ERROR RECOVERY
177  005207  104201  000001                  MOV     #1,R1           ; DEFERRED CALL
178  005211  100651  000022                  MOV     R1,U.MSEC(R5)   ; ONLY TRY TO WRITE ONE SECTOR
179  005213  005215                          BR      13$             ; BRANCH
180  005214  114002                  17$:    CLR     R2              ; NO ERRORS
181  005215  003231                  13$:    BR      JMPRET          ; RETURN TO CALLING PROGRAM
182                                          .DSABL  LSB
186                                          .IF     LE,BUFARA-.
187                                  BUFARA  =       .+1
188                                          .ENDC
189                                          .IF     LE,MAXADR-.
190                                  MAXADR  =       .+1
191                                          .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE READ
005216                                  DMOVLY  R,AREA0
005216  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
000012                          READ    =       AFTWRT+1
                                ;
                                ;       READ READS A BLOCK FROM THE DEVICE
                                ;
                                        .ENABL  LSB
004413  024172                          CALL    DSABLE          ; DISABLE ERROR RECOVERY
004414  104657  000012                  MOV     U.RWTO(R5),R0   ; MOVE READ/WRITE TIMEOUT VALUE TO R0
004416  115407                          INC     R0              ; INCREMENT COUNT
004417  100657  000012                  MOV     R0,U.RWTO(R5)   ; SAVE READ/WRITE TIMEOUT
004421  014531                          BEQ     5$              ; IF ZERO, FIRST TIME -- BRANCH
004422  106207  000002                  CMP     #2,R0           ; SEE IF ATTEMPTED MAX TIMES
004424                                  BCS     11$             ; IF SO, BRANCH
004424  044426                                                                  BCC     .+2
004425  004437                                                                  BR      11$
004426  104141                          MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
004427                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
004427  102201  001000                  BIT     #RTRIES,R1      ; SEE IF RETRIES ENABLED
004431  054531                          BNE     5$              ; IF SO, BRANCH
004432  104651  000046                  MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
004434  102201  000400                  BIT     #REVEC,R1       ; SEE IF READING RCT
004436  054531                          BNE     5$              ; IF SO, GO FOR RETRIES
004437                          11$:    HARDER  26,<S.LETR(R4),U.CBN(R5),U.CBN+1(R5)>
004437  104200  002366  001107                                                  MOV     #ER26,HRQ.04
004442  104640  000005  001110                                                  MOV     S.LETR(R4),HRQ.05
004445  104650  000053  001111                                                  MOV     U.CBN(R5),HRQ.06
004450  104650  000054  001112                                                  MOV     U.CBN+1(R5),HRQ.07
004453  104202  107372                                                          MOV     #26!ERHARD+4000.,R2
004455  104020  001105                                                          MOV     R2,HRQ.02
004457  104200  004457  001104                                                  MOV     #.,HRQ.01
004462  104200  060014  001103                                                  MOV     #ERRMC,HRQ.RQ
004465                          ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5),U.RBN+1(R5)>
004465  104650  000046  001113                                                  MOV     U.PARM(R5),HRQ.08
004470  104200  005472  001114                                                  MOV     #RBNTXT,HRQ.09
004473  104650  000055  001115                                                  MOV     U.RBN(R5),HRQ.10
004476  104650  000056  001116                                                  MOV     U.RBN+1(R5),HRQ.11
004501                          ENDERR  0
004501  114000  002230                                                          CLR     ERRPOS          ; CLEAR THE POSITION
004503  104653  000046                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
004505  103203  000200                  BIC     #RBNBN,R3       ; IF HANDLING AN RBN, CLEAR IT
004507  100653  000046                  MOV     R3,U.PARM(R5)   ; SAVE
004511  104207  000001                  MOV     #1,R0           ; ONE SECTOR HANDLED
004513  100657  000021                  MOV     R0,U.NSEC(R5)   ; SAVE
004515  104207  000001                  MOV     #SETUP,R0       ; SETUP IS NEXT MODULE CALLED
004517  104051                          MOV     R5,R1           ; DELAYED CALL TO SETUP
004520  102203  000400                  BIT     #REVEC,R3       ; SEE IF READING RCT
004522  014536                          BEQ     8$              ; IF NOT, BRANCH
004523  104207  000022                  MOV     #REVCT,R0       ; REVECTOR NEXT MODULE
```

********** OVERLAY MODULE READ

```
     66 004525  104200  177777  002220          MOV     #-1,SCR2        ; MARK SECTOR AS BAD
     67 004530  004535                          BR      4$              ; EXIT
     68 004531  021777                  5$:     CALL    BULDUM          ; BUILD THE DUMMY SDI CONTROL BLOCK
     69 004532  024537                          CALL    RBLOCK          ; READ THE SECTOR
     70 004533  104207  000013                  MOV     #SECCHK,R0      ; SECCHK NEXT MODULE CALLED
     71 004535  114001                  4$:     CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
     72 004536  003231                  8$:     BR      JMPRET          ; RETURN TO RDWRT MODULE
     73                                          .DSABL  LSB
```

```
                                .SBTTL  RBLOCK - READ THE SECTORS
004537                          RBLOCK:
                                ;
                                ;       READ A SECTOR FROM THE DEVICE
                                ;
004537  104307  002232                  MOV     CHAINS,R0       ; POINT TO START OF CHAIN
004541  104652  000025                  MOV     U.MASK(R5),R2   ; R2 HAS SDI INTERCONNECT
004543  060012                          XFC     WAITSI          ; WAIT FOR SECTOR OR INDEX PULSE
004544  115001                          TST     R1              ; SEE IF ERROR OCCURRED
004545  014604                          BEQ     6$
004546                                  DEVFTL  42,<U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
004546  104200  003066  001107                                          MOV     #ER42,HRQ.04
004551  104650  000066  001110                                          MOV     U.CGRP(R5),HRQ.05
004554  104650  000064  001111                                          MOV     U.CCYL(R5),HRQ.06
004557  104650  000065  001112                                          MOV     U.CCYL+1(R5),HRQ.07
004562  104202  047712                                                  MOV     #42!FTLDEV+4000.,R2
004564  104020  001105                                                  MOV     R2,HRQ.02
004566  104200  004566  001104                                          MOV     #.,HRQ.01
004571  104200  060014  001103                                          MOV     #ERRMC,HRQ.RQ
004574                                  ENDERR
004574  104200  000051  001113                                          MOV     #SER22,HRQ.08
004577  104200  000011  002230                                          MOV     #8+1,ERRPOS     ; SET THE POSITION
004602  024207                          CALL    GOSEEK
004603  004640                          BR      5$              ; EXIT
004604                          6$:
004604  060002                          XFC     XREAD           ; WRITE THE SECTOR(S)
004605  100651  000061                  MOV     R1,U.RWER(R5)   ; SAVE ERROR TYPE
004607  024641                          CALL    FNDRER          ; FIND READ ERROR (FILL IN CORRECT BN NUMBERS)
004610  115001                          TST     R1              ; SEE IF ANY ERRORS OCCURRED
004611  054637                          BNE     4$              ; IF SO, BRANCH
004612  104657  000012                  MOV     U.RWTO(R5),R0   ; SEE IF ANY RETRIES
004614  014637                          BEQ     4$              ; IF SO, BRANCH
004615                                  REPSFT  SOFT            ; REPORT SOFT ERROR
004615  104200  000001  001105                                          MOV     #1,HRQ.02
004620  114000  001106                                                  CLR     HRQ.03
004622  114000  001107                                                  CLR     HRQ.04
004624  100467                                                                  MOV R0,-(SP)
004625  104657  000063                                                  MOV     U.UNUM(R5),R0
004627  105657  000050                                                  ADD     U.SUBU(R5),R0
004631  104070  001104                                                  MOV     R0,HRQ.01
004633  104207  060007                                                  MOV     #T4SOFT,R0
004635  021053                                                          CALL    HOSTRQ
004636  104267                                                                  MOV (SP)+,R0
004637  114002                  4$:     CLR     R2              ; NO ERRORS
004640  000000                  5$:     RETURN                  ; RETURN TO CALLING PROGRAM;
```

```
                                .SBTTL  FNDRER - IF ERROR DURING READ, FIND IT'S POSITION IN THE CHAIN
004641                          FNDRER:
                                ;
                                ;       FIND THE FIRST BUFFER THAT IS NOT READ
                                ;
004641                          PUSH    <R1,R4,R5>      ; SAVE REGISTERS
004641  100461                                                          MOV R1,-(SP)
004642  100464                                                          MOV R4,-(SP)
004643  100465                                                          MOV R5,-(SP)
004644  104654  000053          MOV     U.CBN(R5),R4    ; GET LO ORDER BN
004646  104655  000054          MOV     U.CBN+1(R5),R5  ; GET HI BN
004650  104307  002232          MOV     CHAINS,R0       ; R0 POINTS TO FIRST BUFFER
004652  100674  000002  7$:     MOV     R4,RW.LOW(R0)   ; MOVE TO CHAIN
004654  100675  000003          MOV     R5,RW.HI(R0)    ; MOVE TO CHAIN
004656  104171                  MOV     (R0),R1         ; GET STATUS BITS
004657  102201  040000          BIT     #BUFFLG,R1      ; SEE IF THIS BUFFER HAS BEEN READ
004661  014674                  BEQ     8$              ; IF NOT, BRANCH
004662  115001                  TST     R1              ; SEE IF END-OF-LIST
004663  074674                  BMI     8$              ; IF LAST BUFFER, EXIT
004664                          ASSUME  EOC,100000      ; ASSUME READ STOP IS SIGN
004664  103201  170000          BIC     #UNADDR,R1      ; CLEAR UNUSED BITS
004666  104017                  MOV     R1,R0           ; MOVE TO R0
004667  105204  000001          ADD     #1,R4           ; ADD ONE TO LOW ORDER BN
004671  044652                  BCC     7$              ; IF NO CARRY, BRANCH
004672  115405                  INC     R5              ; PRPOGATE CARRY
004673  004652                  BR      7$              ; LOOP
004674                  8$:     POP     <R5,R4,R1>      ; RESTORE REGISTERS
004674  104265                                                          MOV (SP)+,R5
004675  104264                                                          MOV (SP)+,R4
004676  104261                                                          MOV (SP)+,R1
004677  000000                  RETURN                  ; RETURN TO CALLING PROGRAM
                                .IF     LT,BUFARA-.
                        BUFARA  =       .
                                .ENDC
                                .IF     LE,MAXADR-.
                        MAXADR  =       .+1
                                .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE SECCHK - CHECK THAT BUFFER FULL AND ECC
004700                                  DMOVLY  SC,AREA0
004700  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000013                          SECCHK  =       READ+1
                                ;
                                ;       CHECK TO SEE IF THE SECTOR'S FULL, IF NOT, REPORT WHY (UNLESS
                                ;       REVECTOR, WHERE I'LL REVECTOR THE BLOCK)
                                ;       IF THE SECTOR'S FULL, CHECK ECC AND EDC, THEN CALL THE NEXT ROUTINE
                                ;       DEPENDING ON THE OPTIONS IN U.PARM (SELECTED BY USER)
                                ;
                                        .ENABL  LSB
004413  104307  002232                  MOV     CHAINS,R0       ; GET POINTER TO READ CHAIN
004415  104171                          MOV     (R0),R1         ; GET STATUS OF BUFFER
004416                                  ASSUME  RW.STA,0
004416  102201  040000                  BIT     #BUFFLG,R1      ; SEE IF BUFFER FULL
004420  055222                          BNE     4$              ; IF SO, BRANCH
004421  021725                          CALL    BLKCHK          ; SEE IF THIS IS A KNOWN BAD BLOCK
004422  045207                          BCC     3$              ; IF SO, BRANCH
004423  104651  000021                  MOV     U.NSEC(R5),R1   ; GET NUMBER OF SECTORS READ
004425  105651  000024                  ADD     U.CSEC(R5),R1   ; ADD NUMBER OF SECTORS PREVIOUSLY HANDLED
004427  100651  000024                  MOV     R1,U.CSEC(R5)   ; SAVE
004431  104651  000021                  MOV     U.NSEC(R5),R1   ; GET NUMBER OF SECTORS READ
004433  105641  000010                  ADD     S.MEGR(R4),R1   ; ADD TO MEGABITS READ
004435  100641  000010                  MOV     R1,S.MEGR(R4)   ; SAVE
004437  104651  000021                  MOV     U.NSEC(R5),R1   ; GET NUMBER OF SECTORS READ
004441  105651  000053                  ADD     U.CBN(R5),R1    ; ADJUST U.CBN TO SECTOR WITH ERROR
004443  100651  000053                  MOV     R1,U.CBN(R5)    ; SAVE
004445  044453                          BCC     1$              ; IF NO CARRY, BRANCH
004446  104651  000054                  MOV     U.CBN+1(R5),R1  ; GET CURRENT BN
004450  115401                          INC     R1              ; PROPOGATE CARRY
004451  100651  000054                  MOV     R1,U.CBN+1(R5)  ; SAVE
004453  114001                  1$:     CLR     R1              ; TO CLEAR NUMBER OF SECTORS HANDLED
004454  100651  000021                  MOV     R1,U.NSEC(R5)   ; ZERO SECTORS HANDLED
004456  104651  000061                  MOV     U.RWER(R5),R1   ; GET READ/WRITE ERROR
004460  106201  000003                  CMP     #3,R1           ; SEE IF REVECTORED BLOCK
004462  014524                          BEQ     2$              ; IF SO, BRANCH
004463  106201  000002                  CMP     #2,R1           ; SEE IF FAILURE IN DRIVE
004465  054505                          BNE     8$              ; IF NOT, BRANCH
004466                                  SOFTER  20              ; REPORT
004466  104200  002056  001107                                          MOV     #ER20,HRQ.04
004471  104202  147664                                                  MOV     #20!ERSOFT+4000.,R2
004473  104020  001105                                                  MOV     R2,HRQ.02
004475  104200  004475  001104                                          MOV     #.,HRQ.01
004500  104200  060013  001103                                          MOV     #ERRMES,HRQ.RQ
004503  024223                          CALL    GOINIT
004504  005110                          BR      12$             ; BRANCH
004505  106201  000004          8$:     CMP     #4,R1           ; SEE IF HEADER COMPARE FAILURE
004507  054643                          BNE     6$              ; IF NOT, BRANCH
004510  104653  000046                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
004512  104141                          MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
```

```
 87 004513                                          ASSUME  S.PARM,0
 88 004513  102201  020000                          BIT     #DCYLS,R1       ; SEE IF ON DIAGNOSTIC CYLINDERS
 89 004515  054527                                  BNE     9$              ; IF SO, BRANCH
 90 004516  102203  000400                          BIT     #REVEC,R3       ; SEE IF REVECTOR ENABLED
 91 004520  054635                                  BNE     5$              ; IF SO, BRANCH AROUND
 92 004521  102203  000200                          BIT     #RBNBN,R3       ; SEE IF READING RCT OR RBN
 93 004523  054527                                  BNE     9$              ; IF SO, REPORT ERROR
 94 004524  104207  000022          2$:             MOV     #REVCT,R0       ; REVECTOR NEXT MODULE
 95 004526  005224                                  BR      19$             ; EXIT
 96 004527                          9$:             HARDER  19              ; REPORT HEADER COMPARE ERROR
    004527  104200  001754  001107                                          MOV     #ER19,HRQ.04
    004532  104202  107663                                                  MOV     #19!ERHARD+4000.,R2
    004534  104020  001105                                                  MOV     R2,HRQ.02
    004536  104200  004536  001104                                          MOV     #.,HRQ.01
    004541  104200  060014  001103                                          MOV     #ERRMC,HRQ.RQ
 97 004544                                          ERRORC  <S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
    004544  104640  000005  001110                                          MOV     S.LETR(R4),HRQ.05
    004547  104670  000002  001111                                          MOV     RW.LOW(R0),HRQ.06
    004552  104670  000003  001112                                          MOV     RW.HI(R0),HRQ.07
 98 004555                                          ERRORC  <R3,#RBNTXT,U.RBN(R5)>
    004555  104030  001113                                                  MOV     R3,HRQ.08
    004557  104200  005472  001114                                          MOV     #RBNTXT,HRQ.09
    004562  104650  000055  001115                                          MOV     U.RBN(R5),HRQ.10
 99 004565                                          ERRORC  <U.RBN+1(R5),RW.ANG(R0)>
    004565  104650  000056  001116                                          MOV     U.RBN+1(R5),HRQ.11
    004570  104670  000006  001117                                          MOV     RW.ANG(R0),HRQ.12
100 004573                                          ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
    004573  104670  000004  001120                                          MOV     RW.CMD(R0),HRQ.13
    004576  104650  000066  001121                                          MOV     U.CGRP(R5),HRQ.14
    004601  104650  000064  001122                                          MOV     U.CCYL(R5),HRQ.15
    004604  104650  000065  001123                                          MOV     U.CCYL+1(R5),HRQ.16
101 004607                                          ERRORC  <U.LGRP(R5),U.LCYL(R5),U.LCYL+1(R5)>
    004607  104650  000071  001124                                          MOV     U.LGRP(R5),HRQ.17
    004612  104650  000067  001125                                          MOV     U.LCYL(R5),HRQ.18
    004615  104650  000070  001126                                          MOV     U.LCYL+1(R5),HRQ.19
102 004620                                          ENDERR  0
    004620  114000  002230                                                  CLR     ERRPOS          ; CLEAR THE POSITION
103 004622  103203  000200                          BIC     #RBNBN,R3       ; IF HANDLING A RBN, CLEAR IT
104 004624  100653  000046                          MOV     R3,U.PARM(R5)   ; SAVE
105 004626  104201  000001                          MOV     #1,R1           ; ONLY ONE SECTOR HANDLED
106 004630  100651  000021                          MOV     R1,U.NSEC(R5)   ; SAVE
107 004632  104207  000001                          MOV     #SETUP,R0       ; SETUP NEXT MODULE
108 004634  005226                                  BR      18$             ; BRANCH + REPORT, DELAYED CALL
109 004635  104207  000022          5$:             MOV     #REVCT,R0       ; REVECTOR NEXT MODULE
110 004637  104200  177777  002220                  MOV     #-1,SCR2        ; SECTOR MARKED BAD
111 004642  005224                                  BR      19$             ; EXIT, IMMIDATE CALL
112 004643  106201  000052          6$:             CMP     #52,R1          ; SEE IF SERDES OVERRUN ERROR
113 004645  054664                                  BNE     20$             ; IF NOT, BRANCH
114 004646                                          SOFTER  34              ; SERDES OVERRUN
    004646  104200  002540  001107                                          MOV     #ER34,HRQ.04
    004651  104202  147702                                                  MOV     #34!ERSOFT+4000.,R2
    004653  104020  001105                                                  MOV     R2,HRQ.02
    004655  104200  004655  001104                                          MOV     #.,HRQ.01
    004660  104200  060013  001103                                          MOV     #ERRMES,HRQ.RQ
115 004663  005110                                  BR      12$             ; REST OF ERROR MESSAGE
116 004664  106201  000150          20$:            CMP     #150,R1         ; SEE IF DATA SYNC TIMEOUT
117 004666  054705                                  BNE     22$             ; IF NOT, BRANCH
```

```
118 004667                                      SOFTER  36          ; DATA SYNC TIMEOUT
    004667  104200  002612  001107                                          MOV     #ER36,HRQ.04
    004672  104202  147704                                                  MOV     #36!ERSOFT+4000.,R2
    004674  104020  001105                                                  MOV     R2,HRQ.02
    004676  104200  004676  001104                                          MOV     #.,HRQ.01
    004701  104200  060013  001103                                          MOV     #ERRMES,HRQ.RQ
119 004704  005110                                      BR      12$         ; REST OF ERROR MESSAGE
120 004705  106201  000153          22$:    CMP     #153,R1     ; SEE IF POSITIONER ERROR
121 004707  054751                          BNE     23$         ; IF NOT, BRANCH
122 004710                                      REPSFT  ,,SEEK      ; REPORT SEEK ERROR
    004710  114000  001105                                          CLR     HRQ.02
    004712  114000  001106                                          CLR     HRQ.03
    004714  104200  000001  001107                                  MOV     #1,HRQ.04
    004717  100467                                                          MOV R0,-(SP)
    004720  104657  000063                                          MOV     U.UNUM(R5),R0
    004722  105657  000050                                          ADD     U.SUBU(R5),R0
    004724  104070  001104                                          MOV     R0,HRQ.01
    004726  104207  060007                                          MOV     #T4SOFT,R0
    004730  021053                                                  CALL    HOSTRQ
    004731  104267                                                          MOV (SP)+,R0
123 004732                                      SOFTER  45          ; REPORT POSITIONER ERROR
    004732  104200  003162  001107                                  MOV     #ER45,HRQ.04
    004735  104202  147715                                          MOV     #45!ERSOFT+4000.,R2
    004737  104020  001105                                          MOV     R2,HRQ.02
    004741  104200  004741  001104                                  MOV     #.,HRQ.01
    004744  104200  060013  001103                                  MOV     #ERRMES,HRQ.RQ
124 004747  024215                                      CALL    GORCLB      ; RECALIBRATION REQUIRED FOR RECOVERY
125 004750  005110                                      BR      12$         ; REST OF ERROR MESSAGE
126 004751  106201  000213          23$:    CMP     #213,R1     ; SEE IF READ/WRITE READY FAILURE
127 004753  054772                          BNE     24$         ; IF NOT, BRANCH
128 004754                                      SOFTER  37          ; READ/WRITE READY FAILURE
    004754  104200  002634  001107                                  MOV     #ER37,HRQ.04
    004757  104202  147705                                          MOV     #37!ERSOFT+4000.,R2
    004761  104020  001105                                          MOV     R2,HRQ.02
    004763  104200  004763  001104                                  MOV     #.,HRQ.01
    004766  104200  060013  001103                                  MOV     #ERRMES,HRQ.RQ
129 004771  005110                                      BR      12$         ; REST OF ERROR MESSAGE
130 004772  106201  000253          24$:    CMP     #253,R1     ; SEE IF DATA DRIVE OR STATE CLOCK TIMEOUT
131 004774  055014                          BNE     21$         ; IF NOT, BRANCH
132 004775                                      SOFTER  35          ; DRIVE CLOCK TIMEOUT
    004775  104200  002563  001107                                  MOV     #ER35,HRQ.04
    005000  104202  147703                                          MOV     #35!ERSOFT+4000.,R2
    005002  104020  001105                                          MOV     R2,HRQ.02
    005004  104200  005004  001104                                  MOV     #.,HRQ.01
    005007  104200  060013  001103                                  MOV     #ERRMES,HRQ.RQ
133 005012  024223                                      CALL    GOINIT      ; INIT THE DRIVE TO RECOVER
134 005013  005110                                      BR      12$         ; REST OF ERROR MESSAGE
135 005014  106201  000313          21$:    CMP     #313,R1     ; SEE IF RECIEVER READY TIMEOUT
136 005016  055036                          BNE     25$         ; IF NOT, BRANCH
137 005017                                      SOFTER  38          ; RECIEVER READY FAILURE
    005017  104200  002660  001107                                  MOV     #ER38,HRQ.04
    005022  104202  147706                                          MOV     #38!ERSOFT+4000.,R2
    005024  104020  001105                                          MOV     R2,HRQ.02
    005026  104200  005026  001104                                  MOV     #.,HRQ.01
    005031  104200  060013  001103                                  MOV     #ERRMES,HRQ.RQ
138 005034  024223                                      CALL    GOINIT      ; INIT THE DRIVE TO RECOVER
139 005035  005110                                      BR      12$         ; REST OF ERROR MESSAGE
```

```
140 005036  106201  000413          25$:  CMP     #413,R1          ; RTDS FAILURE DURING READ?
141 005040  055057                        BNE     26$              ; IF NOT, BRANCH
142 005041                                SOFTER  64               ; REPORT
    005041  104200  004145  001107                                        MOV     #ER64,HRQ.04
    005044  104202  147740                                                MOV     #64!ERSOFT+4000.,R2
    005046  104020  001105                                                MOV     R2,HRQ.02
    005050  104200  005050  001104                                        MOV     #.,HRQ.01
    005053  104200  060013  001103                                        MOV     #ERRMES,HRQ.RQ
143 005056  005110                        BR      12$              ; BRANCH
144 005057                          26$:  HARDER  69,<#SER36,R1>   ; UNKNOWN ERROR NUMBER
    005057  104200  004220  001107                                        MOV     #ER69,HRQ.04
    005062  104200  004242  001110                                        MOV     #SER36,HRQ.05
    005065  104010  001111                                                MOV     R1,HRQ.06
    005067  104202  107745                                                MOV     #69!ERHARD+4000.,R2
    005071  104020  001105                                                MOV     R2,HRQ.02
    005073  104200  005073  001104                                        MOV     #.,HRQ.01
    005076  104200  060014  001103                                        MOV     #ERRMC,HRQ.RQ
145 005101                                ENDERR
    005101  104200  000051  001112                                        MOV     #SER22,HRQ.07
    005104  104200  000010  002230                                        MOV     #7+1,ERRPOS      ; SET THE POSITION
146 005107  005201                        BR      14$              ; REST OF ERROR MESSAGE
147 005110                          12$:  CERROR  5,#SER19
    005110  104200  000470  001110                                        MOV     #SER19,HRQ.05
148 005113                                ERRORC  <U.RWTO(R5),S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
    005113  104650  000012  001111                                        MOV     U.RWTO(R5),HRQ.06
    005116  104640  000005  001112                                        MOV     S.LETR(R4),HRQ.07
    005121  104670  000002  001113                                        MOV     RW.LOW(R0),HRQ.08
    005124  104670  000003  001114                                        MOV     RW.HI(R0),HRQ.09
149 005127                                ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5)>
    005127  104650  000046  001115                                        MOV     U.PARM(R5),HRQ.10
    005132  104200  005472  001116                                        MOV     #RBNTXT,HRQ.11
    005135  104650  000055  001117                                        MOV     U.RBN(R5),HRQ.12
150 005140                                ERRORC  <U.RBN+1(R5),RW.ANG(R0)>
    005140  104650  000056  001120                                        MOV     U.RBN+1(R5),HRQ.13
    005143  104670  000006  001121                                        MOV     RW.ANG(R0),HRQ.14
151 005146                                ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
    005146  104670  000004  001122                                        MOV     RW.CMD(R0),HRQ.15
    005151  104650  000066  001123                                        MOV     U.CGRP(R5),HRQ.16
    005154  104650  000064  001124                                        MOV     U.CCYL(R5),HRQ.17
    005157  104650  000065  001125                                        MOV     U.CCYL+1(R5),HRQ.18
152 005162                                ERRORC  <U.LGRP(R5),U.LCYL(R5),U.LCYL+1(R5)>
    005162  104650  000071  001126                                        MOV     U.LGRP(R5),HRQ.19
    005165  104650  000067  001127                                        MOV     U.LCYL(R5),HRQ.20
    005170  104650  000070  001130                                        MOV     U.LCYL+1(R5),HRQ.21
153 005173                                ENDERR
    005173  104200  000051  001131                                        MOV     #SER22,HRQ.22
    005176  104200  000027  002230                                        MOV     #22+1,ERRPOS     ; SET THE POSITION
154 005201  024207                  14$:  CALL    GOSEEK           ; SEEK REQUIRED FOR RECOVERY
155 005202  104207  000001                MOV     #1,R0            ; SET UP TO READ ONE SECTOR
156 005204  100657  000022                MOV     R0,U.MSEC(R5)    ; SAVE
157 005206  005226                        BR      18$              ; EXIT
158 005207  104641  000010            3$: MOV     S.MEGR(R4),R1    ; GET NUMBER OF SECTORS READ
159 005211  117401                        DEC     R1               ; READJUST TO ACTUAL SECTORS TESTED
160 005212  100641  000010                MOV     R1,S.MEGR(R4)    ; SAVE
161 005214  104201  100000                MOV     #EOC,R1          ; MOVE END-OF-CHAIN TO R1
162 005216  100171                        MOV     R1,(R0)          ; FLAG THIS BUFFER AS END-OF-CHAIN (FOR LSTMOD)
163 005217  104207  000021                MOV     #LSTMOD,R0       ; LSTMOD NEXT MODULE
```

********** OVERLAY MODULE SECCHK - CHECK THAT BUFFER FULL AND

```
    164 005221  005224                        BR      19$             ; EXIT, NO ERRORS, IMMIDATE CALL
    165 005222  104207  000014          4$:   MOV     #CHKEDC,R0      ; CHKEDC IS NEXT MODULE
    166 005224  114002                  19$:  CLR     R2              ; NO ERRORS
    167 005225  114001                  17$:  CLR     R1              ; IMMIDATE CALL
    168 005226  003231                  18$:  BR      JMPRET          ; RETURN TO SEQUNCR
    169                                       .DSABL  LSB
    173                                       .IF     LE,BUFARA-.
    174                                 BUFARA =      .+1
    175                                       .ENDC
    176                                       .IF     LE,MAXADR-.
    177                                 MAXADR =      .+1
    178                                       .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE CHKEDC - CHECK THAT ECC AND EDC ARE OK
005227                                  DMOVLY  ED,AREA0
005227  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000014                          CHKEDC  =       SECCHK+1
                                ;
                                ;       CHECK THAT THERE IS NO ECC ERROR IN THIS BUFFER, IF NOT, CHECK THE
                                ;       EDC, IF THAT IS INCORRECT YOU HAVE A MAJOR ERROR...
                                ;       IF ECC AND EDC IS OK, GO TO DATA COMPARE (IF REQUESTED)
                                ;
                                        .ENABL  LSB
004413  104307  002232                  MOV     CHAINS,R0       ; R0 POINTS AT READ CHAIN LINK BEING TESTED
004415  104171                          MOV     (R0),R1         ; R1 IS BUFFER STATUS
004416                                  ASSUME  RW.STAT,0       ; ASSUME STATUS IS IN FIRST WORD
004416  102201  010000                  BIT     #ECCFLG,R1      ; SEE IF ECC ERROR IN BUFFER
004420  014547                          BEQ     11$             ; IF NOT, CHECK EDC
004421  104653  000046                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
004423  102203  000400                  BIT     #REVEC,R3       ; SEE IF FINDING A REVECTOR
004425  054445                          BNE     2$              ; IF SO, BRANCH
004426  021725                          CALL    BLKCHK          ; SEE IF THIS IS A KNOWN BAD BLOCK
004427                                  BCS     5$              ; IF NOT, BRANCH
004427  044431                                                                  BCC     .+2
004430  004441                                                                  BR      5$
004431  104641  000010                  MOV     S.MEGR(R4),R1   ; GET NUMBER OF SECTORS READ
004433  117401                          DEC     R1              ; READJUST TO ACTUAL SECTORS HANDLED
004434  100641  000010                  MOV     R1,S.MEGR(R4)   ; SAVE
004436  104207  000021                  MOV     #LSTMOD,R0      ; LAST MODULE IS NEXT
004440  004776                          BR      19$             ; EXIT, NO ERRORS, IMMIDATE CALL
004441  104141                  5$:     MOV     (R4),R1         ; GET UNIT PARAMETERS
004442                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
004442  102201  010000                  BIT     #ECCCHK,R1      ; SEE IF ECC CORRECTION IS REQUESTED
004444  014450                          BEQ     6$              ; IF NOT, BRANCH
004445  104207  000015          2$:     MOV     #CHKECC,R0      ; CHKECC NEXT MODULE
004447  004776                          BR      19$             ; EXIT, NO ERRORS, IMMIDATE CALL
004450                          6$:     SOFTER  7,<#SER21,U.RRTY(R5),U.ELEV(R5)>
004450  104200  000615  001107                                                  MOV     #ER7,HRQ.04
004453  104200  000661  001110                                                  MOV     #SER21,HRQ.05
004456  104650  000010  001111                                                  MOV     U.RRTY(R5),HRQ.06
004461  104650  000027  001112                                                  MOV     U.ELEV(R5),HRQ.07
004464  104202  147647                                                          MOV     #7!ERSOFT+4000.,R2
004466  104020  001105                                                          MOV     R2,HRQ.02
004470  104200  004470  001104                                                  MOV     #.,HRQ.01
004473  104200  060013  001103                                                  MOV     #ERRMES,HRQ.RQ
004476                                  ERRORC  <S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
004476  104640  000005  001113                                                  MOV     S.LETR(R4),HRQ.08
004501  104670  000002  001114                                                  MOV     RW.LOW(R0),HRQ.09
004504  104670  000003  001115                                                  MOV     RW.HI(R0),HRQ.10
004507                                  ERRORC  <U.PARM(R5),#RBNTXT>
004507  104650  000046  001116                                                  MOV     U.PARM(R5),HRQ.11
004512  104200  005472  001117                                                  MOV     #RBNTXT,HRQ.12
004515                                  ERRORC  <U.RBN(R5),U.RBN+1(R5),RW.ANG(R0)>
```

```
********** OVERLAY MODULE CHKEDC - CHECK THAT ECC AND EDC ARE

        004515  104650  000055  001120                                          MOV     U.RBN(R5),HRQ.13
        004520  104650  000056  001121                                          MOV     U.RBN+1(R5),HRQ.14
        004523  104670  000006  001122                                          MOV     RW.ANG(R0),HRQ.15
     77 004526                          ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5)>
        004526  104670  000004  001123                                          MOV     RW.CMD(R0),HRQ.16
        004531  104650  000066  001124                                          MOV     U.CGRP(R5),HRQ.17
        004534  104650  000064  001125                                          MOV     U.CCYL(R5),HRQ.18
     78 004537                          ERRORC  U.CCYL+1(R5)
        004537  104650  000065  001126                                          MOV     U.CCYL+1(R5),HRQ.19
     79 004542                          ENDERR  0
        004542  114000  002230                                                  CLR     ERRPOS          ; CLEAR THE POSITION
     80 004544  104651  000046          MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
     81 004546  004742                  BR      10$             ; EXIT
     82 004547  104677  000001  11$:    MOV     RW.BUF(R0),R0   ; R0 NOW POINTS AT BUFFER
     83 004551  021146                  CALL    CMPEDC          ; COMPUTE THE EDC
     84 004552  104651  000046          MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
     85 004554  106672  000400          CMP     BF.EDC(R0),R2   ; SEE IF MATCH
     86 004556  054632                  BNE     14$             ; IF NOT, BRANCH
     87 004557  104657  000010          MOV     U.RRTY(R5),R0   ; GET RETRIES
     88 004561  054567                  BNE     23$             ; IF RETRIES WERE USED, REPORT SOFT ERROR
     89 004562  104657  000027          MOV     U.ELEV(R5),R0   ; GET ERROR LEVEL
     90 004564  106657  000031          CMP     U.MLEV(R5),R0   ; SEE IF ANY ERROR LEVELS WERE USED
     91 004566  014611                  BEQ     22$             ; IF NOT, BRANCH
     92 004567                  23$:    REPSFT  SOFT            ; REPORT SOFT ERROR
        004567  104200  000001  001105                                          MOV     #1,HRQ.02
        004572  114000  001106                                                  CLR     HRQ.03
        004574  114000  001107                                                  CLR     HRQ.04
        004576  100467                                                                  MOV R0,-(SP)
        004577  104657  000063                                                  MOV     U.UNUM(R5),R0
        004601  105657  000050                                                  ADD     U.SUBU(R5),R0
        004603  104070  001104                                                  MOV     R0,HRQ.01
        004605  104207  060007                                                  MOV     #T4SOFT,R0
        004607  021053                                                          CALL    HOSTRQ
        004610  104267                                                                  MOV (SP)+,R0
     93 004611  114000  002220  22$:    CLR     SCR2            ; FLAG AS GOOD BLOCK
     94 004613  102201  000400          BIT     #REVEC,R1       ; SEE IF REVECTOR IN PROGRESS
     95 004615  014621                  BEQ     1$              ; IF NOT, BRANCH
     96 004616  104207  000022          MOV     #REVCT,R0       ; REVECTOR NEXT MODULE
     97 004620  004776                  BR      19$             ; EXIT
     98 004621  102201  000002  1$:     BIT     #DATCMP,R1      ; SEE IF DATA COMPARE REQUESTED
     99 004623  054627                  BNE     12$             ; IF SO, BRANCH
    100 004624  104207  000021          MOV     #LSTMOD,R0      ; LSTMOD NEXT MODULE
    101 004626  004776                  BR      19$             ; EXIT, NO ERRORS, IMMIDATE CALL
    102 004627  104207  000020  12$:    MOV     #CMPDAT,R0      ; DATA COMPARE NEXT MODULE
    103 004631  004776                  BR      19$             ; EXIT, NO ERRORS, IMMIDATE CALL
    104 004632  104307  002232  14$:    MOV     CHAINS,R0
    105 004634  104020  002220          MOV     R2,SCR2         ; SAVE CALCULATED EDC
    106 004636  104673  000001          MOV     RW.BUF(R0),R3
    107 004640                          SOFTER  24,<#SER21,U.RRTY(R5),U.ELEV(R5)>
        004640  104200  002272  001107                                          MOV     #ER24,HRQ.04
        004643  104200  000661  001110                                          MOV     #SER21,HRQ.05
        004646  104650  000010  001111                                          MOV     U.RRTY(R5),HRQ.06
        004651  104650  000027  001112                                          MOV     U.ELEV(R5),HRQ.07
        004654  104202  147670                                                  MOV     #24!ERSOFT+4000.,R2
        004656  104020  001105                                                  MOV     R2,HRQ.02
        004660  104200  004660  001104                                          MOV     #.,HRQ.01
        004663  104200  060013  001103                                          MOV     #ERRMES,HRQ.RQ
```

```
                                                ERRORC  <S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
004666                                                          MOV     S.LETR(R4),HRQ.08
004671                                                          MOV     RW.LOW(R0),HRQ.09
004674                                                          MOV     RW.HI(R0),HRQ.10
                                                ERRORC  <U.PARM(R5),#RBNTXT>
004677                                                          MOV     U.PARM(R5),HRQ.11
004702                                                          MOV     #RBNTXT,HRQ.12
                                                ERRORC  <U.RBN(R5),U.RBN+1(R5),RW.ANG(R0)>
004705                                                          MOV     U.RBN(R5),HRQ.13
004710                                                          MOV     U.RBN+1(R5),HRQ.14
004713                                                          MOV     RW.ANG(R0),HRQ.15
                                                ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5)>
004716                                                          MOV     RW.CMD(R0),HRQ.16
004721                                                          MOV     U.CGRP(R5),HRQ.17
004724                                                          MOV     U.CCYL(R5),HRQ.18
                                                ERRORC  <U.CCYL+1(R5),SCR2,BF.EDC(R3)>
004727                                                          MOV     U.CCYL+1(R5),HRQ.19
004732                                                          MOV     SCR2,HRQ.20
004735                                                          MOV     BF.EDC(R3),HRQ.21
                                                ENDERR  0
004740                                                          CLR     ERRPOS          ; CLEAR THE POSITION
004742  104200  177777  002220      10$:      MOV     #-1,SCR2        ; FLAG AS BLOCK BAD
004745  102201  000400                        BIT     #REVEC,R1       ; SEE IF REVECTOR IN PROGRESS
004747  054754                              BNE     13$             ; IF SO, RETRY
004750  104147                      15$:    MOV     (R4),R0         ; GET SUBUNIT PARAMETERS
004751                                      ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
004751  102207  001000                        BIT     #RTRIES,R0      ; SEE IF RETRIES ARE ENABLED
004753  014757                              BEQ     16$             ; IF NOT, BRANCH
004754  104207  000016              13$:    MOV     #ERCOV,R0       ; ERROR RECOVERY IS NEXT MODULE
004756  004777                              BR      20$             ; EXIT, IMMIDATE CALL
004757  103200  040000  001105    16$:      BIC     #C2HARD,HRQ.02  ; MAKE A HARD ERROR
004762  104200  060014  001103              MOV     #ERRMC,HRQ.RQ   ; COUNT ERROR
004765  102201  000002                        BIT     #DATCMP,R1      ; SEE IF DATA COMPARE ENABLED
004767  014773                              BEQ     17$             ; IF NOT, BRANCH
004770  104207  000020                      MOV     #CMPDAT,R0      ; DATA COMPARE IS NEXT MODULE
004772  004777                              BR      20$             ; EXIT, IMMIDATE CALL
004773  104207  000021              17$:    MOV     #LSTMOD,R0      ; LSTMOD IS NEXT MODULE
004775  004777                              BR      20$             ; EXIT, IMMIDATE CALL
004776  114002                      19$:    CLR     R2              ; NO ERRORS
004777  114001                      20$:    CLR     R1              ; IMMIDATE CALL
005000  003231                              BR      JMPRET          ; RETURN TO SEQUNCR
                                            .DSABL  LSB
                                            .IF     LE,BUFARA-.
                                    BUFARA  =       .+1
                                            .ENDC

                                            .IF     LE,MAXADR-.
                                    MAXADR  =       .+1
                                            .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE CHKECC - CORRECT BUFFER READ USING ECC
                                        DMOVLY  CK,AREA0
                                        .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000015                          CHKECC  =       CHKEDC+1        ; DATA CHECK OVERLAY
                                ;
                                ;       SEE IF ECC ERROR WAS DECTECTED
                                ;
                                        .ENABL  LSB
004413  104307  002232                  MOV     CHAINS,R0       ; R0 POINTS TO CHAIN
004415                                  PUSH    R0              ; SAVE POINTER TO LINK
004415  100467                                                                  MOV R0,-(SP)
004416  104200  177777  002220        1$:     MOV     #-1,SCR2        ; ASSUME BUFFER BAD
004421  060015                          XFC     ECC             ; APPLY ECC CORRECTION
004422  115001                          TST     R1              ; SEE IF CORRECTION WORKED
004423  014443                          BEQ     1$              ; IF SO, BRANCH
004424                                  POP     R0              ; RESTORE POINTER TO LINK
004424  104267                                                                  MOV (SP)+,R0
004425                                  SOFTER  8
004425  104200  000631  001107                                          MOV     #ER8,HRQ.04
004430  104202  147650                                                  MOV     #8!ERSOFT+4000.,R2
004432  104020  001105                                                  MOV     R2,HRQ.02
004434  104200  004434  001104                                          MOV     #.,HRQ.01
004437  104200  060013  001103                                          MOV     #ERRMES,HRQ.RQ
004442  004465                          BR      8$              ; BRANCH
004443  106657  000032          1$:     CMP     U.ECCT(R5),R0   ; SEE IF CORRECTIONS = TO OR EXCEED THRESHOLD
004445  014447                          BEQ     6$              ; IF =, BRANCH
004446  034545                          BPL     2$              ; IF NOT, BRANCH
004447                          6$:     POP     R0              ; RESTORE POINTER TO LINK
004447  104267                                                                  MOV (SP)+,R0
004450                                  SOFTER  9
004450  104200  000745  001107                                          MOV     #ER9,HRQ.04
004453  104202  147651                                                  MOV     #9!ERSOFT+4000.,R2
004455  104020  001105                                                  MOV     R2,HRQ.02
004457  104200  004457  001104                                          MOV     #.,HRQ.01
004462  104200  060013  001103                                          MOV     #ERRMES,HRQ.RQ
004465                          8$:     ERRORC  <#SER21,U.RRTY(R5),U.ELEV(R5)>
004465  104200  000661  001110                                          MOV     #SER21,HRQ.05
004470  104650  000010  001111                                          MOV     U.RRTY(R5),HRQ.06
004473  104650  000027  001112                                          MOV     U.ELEV(R5),HRQ.07
004476                                  ERRORC  <S.LETR(R4),RW.LOW(R0),RW.HI(R0)>
004476  104640  000005  001113                                          MOV     S.LETR(R4),HRQ.08
004501  104670  000002  001114                                          MOV     RW.LOW(R0),HRQ.09
004504  104670  000003  001115                                          MOV     RW.HI(R0),HRQ.10
004507                                  ERRORC  <U.PARM(R5),#RBNTXT>
004507  104650  000046  001116                                          MOV     U.PARM(R5),HRQ.11
004512  104200  005472  001117                                          MOV     #RBNTXT,HRQ.12
004515                                  ERRORC  <U.RBN(R5),U.RBN+1(R5),RW.ANG(R0)>
004515  104650  000055  001120                                          MOV     U.RBN(R5),HRQ.13
004520  104650  000056  001121                                          MOV     U.RBN+1(R5),HRQ.14
004523  104670  000006  001122                                          MOV     RW.ANG(R0),HRQ.15
```

```
    64 004526                                    ERRORC  <RW.CMD(R0),U.CGRP(R5),U.CCYL(R5)>
       004526  104670  000004  001123                                        MOV     RW.CMD(R0),HRQ.16
       004531  104650  000066  001124                                        MOV     U.CGRP(R5),HRQ.17
       004534  104650  000064  001125                                        MOV     U.CCYL(R5),HRQ.18
    65 004537                                    ERRORC  U.CCYL+1(R5)
       004537  104650  000065  001126                                        MOV     U.CCYL+1(R5),HRQ.19
    66 004542                                    ENDERR  0
       004542  114000  002230                                                CLR     ERRPOS          ; CLEAR THE POSITION
    67 004544  004763                            BR      11$             ; BRANCH
    68 004545                            2$:     POP     R3              ; RESTORE POINTER TO BUFFER
       004545  104263                                                                MOV (SP)+,R3
    69 004546  104637  000001                    MOV     RW.BUF(R3),R0   ; R0 POINTS TO BUFFER
    70 004550  021146                            CALL    CMPEDC          ; COMPUTE EDC VALUE
    71 004551  106672  000400                    CMP     BF.EDC(R0),R2   ; SEE IF EDC VALUE MATCHES
    72 004553  054657                            BNE     3$              ; IF NOT, BRANCH
    73 004554  104200  000001  001106            MOV     #1,HRQ.03       ; REPORT ECC CORRECTION
    74 004557  104657  000010                    MOV     U.RRTY(R5),R0   ; GET TIMEOUT
    75 004561  054571                            BNE     23$             ; IF RETRIES WERE USED, REPORT SOFT ERROR
    76 004562  104657  000027                    MOV     U.ELEV(R5),R0   ; GET ERROR LEVEL
    77 004564  114000  001105                    CLR     HRQ.02          ; ASSUME NO SOFT ERROR
    78 004566  106657  000031                    CMP     U.MLEV(R5),R0   ; SEE IF ANY ERROR LEVELS WERE USED
    79 004570  014573                            BEQ     22$             ; IF NOT, BRANCH
    80 004571  115400  001105            23$:    INC     HRQ.02          ; REPORT SOFT ERROR
    81 004573                            22$:    PUSH    R0              ; SAVE R0
       004573  100467                                                                MOV R0,-(SP)
    82 004574  104657  000063                    MOV     U.UNUM(R5),R0   ; GET STARTING SUBUNIT NUMBER
    83 004576  105657  000050                    ADD     U.SUBU(R5),R0   ; ADD OFFSET
    84 004600  104070  001104                    MOV     R0,HRQ.01       ; MOVE TO OUTPUT BUFFER
    85 004602  104207  060007                    MOV     #T4SOFT,R0      ; REPORT REQUEST
    86 004604  021053                            CALL    HOSTRQ          ; SEND TO HOST
    87 004605                                    POP     R0              ; RESTORE R0
       004605  104267                                                                MOV (SP)+,R0
    88 004606  114000  002220                    CLR     SCR2            ; BUFFER GOOD
    89 004610                                    MSSG    3,<S.LETR(R4),RW.LOW(R3),RW.HI(R3),RW.ANG(R3),RW.CMD(R3),U.CGRP(R5),U.CCYL(R
       004610  104200  005616  001105                                        MOV     #MS3,HRQ.02
       004613  104640  000005  001106                                        MOV     S.LETR(R4),HRQ.03
       004616  104630  000002  001107                                        MOV     RW.LOW(R3),HRQ.04
       004621  104630  000003  001110                                        MOV     RW.HI(R3),HRQ.05
       004624  104630  000006  001111                                        MOV     RW.ANG(R3),HRQ.06
       004627  104630  000004  001112                                        MOV     RW.CMD(R3),HRQ.07
       004632  104650  000066  001113                                        MOV     U.CGRP(R5),HRQ.08
       004635  104650  000064  001114                                        MOV     U.CCYL(R5),HRQ.09
       004640  104650  000065  001115                                        MOV     U.CCYL+1(R5),HRQ.10
       004643  100467                                                                MOV R0,-(SP)
       004644  104650  000063  001104                                        MOV     U.UNUM(R5),HRQ.01
       004647  105650  000050  001104                                        ADD     U.SUBU(R5),HRQ.01
       004652  104207  060015                                                MOV     #MESSAG,R0
       004654  021053                                                        CALL    HOSTRQ
       004655  104267                                                                MOV (SP)+,R0
    90 004656  005006                            BR      5$              ; EXIT
    91 004657  104020  002230            3$:     MOV     R2,ERRPOS       ; SAVE THE COMPUTED EDC
    92 004661                                    SOFTER  10,<#SER21,U.RRTY(R5),U.ELEV(R5)>
       004661  104200  000770  001107                                        MOV     #ER10,HRQ.04
       004664  104200  000661  001110                                        MOV     #SER21,HRQ.05
       004667  104650  000010  001111                                        MOV     U.RRTY(R5),HRQ.06
       004672  104650  000027  001112                                        MOV     U.ELEV(R5),HRQ.07
       004675  104202  147652                                                MOV     #10!ERSOFT+4000.,R2
```

```
                                                                MOV     R2,HRQ.02
                                                                MOV     #.,HRQ.01
                                                                MOV     #ERRMES,HRQ.RQ
 93 004707                                      ERRORC  <S.LETR(R4),RW.LOW(R3),RW.HI(R3)>
    004707  104640  000005  001113                              MOV     S.LETR(R4),HRQ.08
    004712  104630  000002  001114                              MOV     RW.LOW(R3),HRQ.09
    004715  104630  000003  001115                              MOV     RW.HI(R3),HRQ.10
 94 004720                                      ERRORC  <U.PARM(R5),#RBNTXT>
    004720  104650  000046  001116                              MOV     U.PARM(R5),HRQ.11
    004723  104200  005472  001117                              MOV     #RBNTXT,HRQ.12
 95 004726                                      ERRORC  <U.RBN(R5),U.RBN+1(R5),RW.ANG(R3)>
    004726  104650  000055  001120                              MOV     U.RBN(R5),HRQ.13
    004731  104650  000056  001121                              MOV     U.RBN+1(R5),HRQ.14
    004734  104630  000006  001122                              MOV     RW.ANG(R3),HRQ.15
 96 004737                                      ERRORC  <RW.CMD(R3),U.CGRP(R5),U.CCYL(R5)>
    004737  104630  000004  001123                              MOV     RW.CMD(R3),HRQ.16
    004742  104650  000066  001124                              MOV     U.CGRP(R5),HRQ.17
    004745  104650  000064  001125                              MOV     U.CCYL(R5),HRQ.18
 97 004750                                      ERRORC  <U.CCYL+1(R5),ERRPOS,BF.EDC(R0)>        ; REPORT ERROR
    004750  104650  000065  001126                              MOV     U.CCYL+1(R5),HRQ.19
    004753  104300  002230  001127                              MOV     ERRPOS,HRQ.20
    004756  104670  000400  001130                              MOV     BF.EDC(R0),HRQ.21
 98 004761                                      ENDERR  0
    004761  114000  002230                                      CLR     ERRPOS          ; CLEAR THE POSITION
 99 004763  104143                  11$:        MOV     (R4),R3         ; GET SUBUNIT PARAMETERS
100 004764                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
101 004764  102203  001000          BIT     #RTRIES,R3      ; SEE IF RETRIES ARE ENABLED
102 004766  055003                  BNE     4$              ; IF SO, BRANCH
103 004767  104653  000046          MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
104 004771  102203  000400          BIT     #REVEC,R3       ; SEE IF REVECTOR IN PROGRESS
105 004773  055003                  BNE     4$              ; IF SO, BRANCH
106 004774  103200  040000  001105  BIC     #C2HARD,HRQ.02  ; MAKE SOFT ERROR A HARD ERROR
107 004777  104200  060014  001103  MOV     #ERRMC,HRQ.RQ   ; COUNT ERROR
108 005002  005017                  BR      10$             ; BRANCH
109 005003  104207  000016          4$:     MOV     #ERCOV,R0       ; ERROR RECOVERY IS NEXT MODULE
110 005005  005027                  BR      7$              ; BRANCH
111 005006  114002                  5$:     CLR     R2              ; NO ERRORS
112 005007  104653  000046                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
113 005011  102203  000400                  BIT     #REVEC,R3       ; SEE IF FINDING A REVECTOR
114 005013  015017                          BEQ     10$             ; IF NOT, BRANCH
115 005014  104207  000022                  MOV     #REVCT,R0       ; REVECTOR NEXT MODULE
116 005016  005027                          BR      7$              ; BRANCH
117 005017  102203  000002          10$:    BIT     #DATCMP,R3      ; SEE IF DATA COMPARE IS REQUESTED
118 005021  015025                          BEQ     9$              ; IF NOT, BRANCH
119 005022  104207  000020                  MOV     #CMPDAT R0      ; DATA COMPARE IS NEXT MODULE
120 005024  005027                          BR      7$              ; BRANCH
121 005025  104207  000021          9$:     MOV     #LSTMOD,R0      ; GO TO LAST MODULE
122 005027  114001                  7$:     CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
123 005030  003231                          BR      JMPRET          ; RETURN
124                                         .DSABL  LSB
128                                         .IF     LE,BUFARA-.
129                                 BUFARA  =       .+1
130                                         .ENDC
131                                         .IF     LE,MAXADR-.
132                                 MAXADR  =       .+1
133                                         .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE ERCOV - DATA ERROR RETRIES AND LEVELS
                                        DMOVLY  EC,AREA0
005031  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
000016                          ERCOV   =       CHKECC+1
                                ;
                                ;       ERROR RECOVERY AND RETRIES
                                ;
                                        .ENABL  LSB
004413  024163                          CALL    ENABLE          ; ENABLE ERROR RECOVERY
004414  104657  000010                  MOV     U.RRTY(R5),R0   ; GET NUMBER OF RETRIES ALLREADY ATTEMPTED
004416  054455                          BNE     1$              ; IF NOT ZERO (FIRST TIME) BRANCH
004417  104651  000021                  MOV     U.NSEC(R5),R1   ; GET NUMBER OF SECTORS READ
004421  105651  000024                  ADD     U.CSEC(R5),R1   ; ADD NUMBER OF SECTORS PREVIOUSLY HANDLED
004423  100651  000024                  MOV     R1,U.CSEC(R5)   ; SAVE
004425  104651  000021                  MOV     U.NSEC(R5),R1   ; GET NUMBER OF SECTORS READ
004427  105641  000010                  ADD     S.MEGR(R4),R1   ; ADD TO MEGABITS WRITTEN
004431  100641  000010                  MOV     R1,S.MEGR(R4)   ; SAVE
004433  104651  000021                  MOV     U.NSEC(R5),R1   ; GET NUMBER OF SECTORS READ
004435  114003                          CLR     R3              ; TO SETUP VALUES
004436  100653  000021                  MOV     R3,U.NSEC(R5)   ; NO SECTORS READ
004440  115403                          INC     R3              ; MAKE R3 ONE
004441  100653  000022                  MOV     R3,U.MSEC(R5)   ; ONLY READ ONE SECTOR
004443  105651  000053                  ADD     U.CBN(R5),R1    ; ADJUST U.CBN TO SECTOR WITH ERROR
004445  100651  000053                  MOV     R1,U.CBN(R5)    ; SAVE
004447  044455                          BCC     1$              ; IF NO CARRY, BRANCH
004450  104651  000054                  MOV     U.CBN+1(R5),R1  ; GET CURRENT BN
004452  115401                          INC     R1              ; PROPOGATE CARRY
004453  100651  000054                  MOV     R1,U.CBN+1(R5)  ; SAVE
004455  106657  000030          1$:     CMP     U.RTRY(R5),R0   ; COMPARE MAXIMUM COUNT WITH RETRIES ATTEMPTED
004457  014473                          BEQ     NLEV            ; IF RETRIES EXHAUSTED, BRANCH
004460  115407                          INC     R0              ; INCREMENT RETRY COUNT
004461  100657  000010                  MOV     R0,U.RRTY(R5)   ; SAVE TIMEOUT
004463  104207  177777                  MOV     #-1,R0          ; START READ RETRIES AT ZERO
004465  100657  000012                  MOV     R0,U.RWTO(R5)   ; SAVE
004467  104207  000007                  MOV     #BUILDP,R0      ; BUILDP IS NEXT MODULE
004471  104051                          MOV     R5,R1           ; DELAYED CALL TO READ
004472  004504                          BR      EREXT           ; BRANCH
004473  104207  000017          NLEV:   MOV     #NEWLEV,R0      ; NEW LEVEL OF ERROR RECOVERY WILL BE TRIED
004475  104653  000047                  MOV     U.RCOV(R5),R3   ; GET ERROR RECOVERY PARAMETERS
004477  101203  010000                  BIS     #NXTLEV,R3      ; GO TO NEXT LEVEL
004501  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
004503  114001                          CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
004504  114002          EREXT:  CLR     R2              ; NO ERRORS
004505  003231                          BR      JMPRET          ; RETURN TO SEQNCR
                                        .DSABL  LSB
                                        .IF     LE,BUFARA-.
                                BUFARA  =       .+1
                                        .ENDC
                                        .IF     LE,MAXADR-.
                                MAXADR  =       .+1
```

```
     77                                          .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE NEWLEV - SEND DRIVE ERROR LEVEL
                                        DMOVLY  NL,AREA0
                                        .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000017                          NEWLEV  =       ERCOV+1
                                ;
                                ;       INITIATE A NEW LEVEL OF ERROR RECOVERY
                                ;
                                        .ENABL  LSB
004413  104653  000047                  MOV     U.RCOV(R5),R3   ; GET RECOVERY WORD
004415  102203  010000                  BIT     #NXTLEV,R3      ; GO TO NEXT LEVEL?
004417  054424                          BNE     3$              ; IF SO, BRANCH
004420  104657  000027                  MOV     U.ELEV(R5),R0   ; GET CURRENT ERROR RECOVERY LEVEL
004422  115407                          INC     R0              ; GO UP ONE LEVEL
004423  004531                          BR      LEVNZR
004424  104657  000027          3$:     MOV     U.ELEV(R5),R0   ; GET CURRENT ERROR RECOVERY LEVEL
004426  054531                          BNE     LEVNZR          ; IF NON-ZERO, BRANCH
004427  024172                          CALL    DSABLE          ; DISABLE ERROR RECOVERY
004430                                  HARDER  11,<S.LETR(R4),U.CBN(R5),U.CBN+1(R5)>
004430  104200  001044  001107                                          MOV     #ER11,HRQ.04
004433  104640  000005  001110                                          MOV     S.LETR(R4),HRQ.05
004436  104650  000053  001111                                          MOV     U.CBN(R5),HRQ.06
004441  104650  000054  001112                                          MOV     U.CBN+1(R5),HRQ.07
004444  104202  107653                                                  MOV     #11!ERHARD+4000.,R2
004446  104020  001105                                                  MOV     R2,HRQ.02
004450  104200  004450  001104                                          MOV     #.,HRQ.01
004453  104200  060014  001103                                          MOV     #ERRMC,HRQ.RQ
004456                                  ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5),U.RBN+1(R5),U.CGRP(R5)>
004456  104650  000046  001113                                          MOV     U.PARM(R5),HRQ.08
004461  104200  005472  001114                                          MOV     #RBNTXT,HRQ.09
004464  104650  000055  001115                                          MOV     U.RBN(R5),HRQ.10
004467  104650  000056  001116                                          MOV     U.RBN+1(R5),HRQ.11
004472  104650  000066  001117                                          MOV     U.CGRP(R5),HRQ.12
004475                                  ERRORC  <U.CCYL(R5),U.CCYL+1(R5)>
004475  104650  000064  001120                                          MOV     U.CCYL(R5),HRQ.13
004500  104650  000065  001121                                          MOV     U.CCYL+1(R5),HRQ.14
004503                                  ENDERR  0
004503  114000  002230                                                  CLR     ERRPOS          ; CLEAR THE POSITION
004505  104653  000046                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
004507  104200  177777  002220          MOV     #-1,SCR2        ; FLAG AS BAD SECTOR
004512  102203  000400                  BIT     #REVEC,R3       ; SEE IF FINDING A REVECTORED SECTOR
004514  014520                          BEQ     1$              ; IF NOT, BRANCH
004515  104207  000022                  MOV     #REVCT,R0       ; GO TO REVCT ROUTINE NEXT
004517  004571                          BR      NEWEXT          ; BRANCH TO EXIT
004520  102203  000002          1$:     BIT     #DATCMP,R3      ; SEE IF DATA COMPARE REQUESTED
004522  014526                          BEQ     2$              ; IF NOT, BRANCH
004523  104207  000020                  MOV     #CMPDAT,R0      ; COMPARE DATA NEXT MODULE
004525  004571                          BR      NEWEXT          ; EXIT
004526  104207  000021          2$:     MOV     #LSTMOD,R0      ; LAST MODULE NEXT MODULE
004530  004571                          BR      NEWEXT          ; EXIT
                                        .DSABL  LSB
```

```
     65 004531  104070  001677          LEVNZR: MOV     R0,ERRLEV       ; MOVE LEVEL TO SDI COMMAND
     66 004533  104203  001643                  MOV     #CR.ERR,R3      ; POINT TO ERROR RECOVERY COMMAND
     67 004535  021173                          CALL    TALK            ; SEND SDI INTERCHANGE
     68 004536  115002                          TST     R2              ; SEE IF AN ERROR OCCURRED
     69 004537  014544                          BEQ     1$              ; IF NOT, BRANCH
     70 004540                                  CERROR  5,#SER4         ; REPORT ERROR
        004540  104200  004726  001110                                                  MOV     #SER4,HRQ.05
     71 004543  004572                          BR      SNDEX           ; BRANCH
     72 004544  104657  000047          1$:     MOV     U.RCOV(R5),R0   ; GET RECOVERY PARAMETERS
     73 004546  102207  010000                  BIT     #NXTLEV,R0      ; DID THIS GO TO THE NEXT LEVEL?
     74 004550  014564                          BEQ     2$              ; IF NOT, BRANCH
     75 004551  103207  010000                  BIC     #NXTLEV,R0      ; CLEAR NEXT LEVEL BIT
     76 004553  101207  020000                  BIS     #LEVUSD,R0      ; FLAG AS LEVELS USED
     77 004555  100657  000047                  MOV     R0,U.RCOV(R5)   ; SAVE
     78 004557  104657  000027                  MOV     U.ELEV(R5),R0   ; GET ERROR RECOVERY LEVEL
     79 004561  117407                          DEC     R0              ; DECREMENT
     80 004562  100657  000027                  MOV     R0,U.ELEV(R5)   ; SAVE
     81 004564  104207  000007          2$:     MOV     #BUILDP,R0      ; BUILDP IS NEXT MODULE CALLED
     82 004566  114002                          CLR     R2              ; NO ERRORS
     83 004567  100652  000010                  MOV     R2,U.RRTY(R5)   ; START RETRIES AT 0
     84 004571  114001                  NEWEXT: CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
     85 004572  003231                  SNDEX:  BR      JMPRET          ; RETURN TO SEQUENCER
     89                                          .IF     LE,BUFARA-.
     90                                  BUFARA  =       .+1
     91                                          .ENDC
     92                                          .IF     LE,MAXADR-.
     93                                  MAXADR  =       .+1
     94                                          .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE CMPDAT - DATA COMPARISON ON READ BUFFER(S)
004573                                  DMOVLY  CD,AREA0
004573  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000020                          CMPDAT  =       NEWLEV+1
                                ;
                                ;       CMPDAT COMPARES THE PATTERN WITH THAT READ FROM THE SECTOR
                                ;
                                        .ENABL  LSB
004413                                  PUSH    <R4,R5>         ; SAVE R4,R5
004413  100464                                                                  MOV R4,-(SP)
004414  100465                                                                  MOV R5,-(SP)
004415  104307  002232                 MOV     CHAINS,R0       ; R0 POINTS TO LINK
004417  104677  000001                 MOV     RW.BUF(R0),R0   ; R0 POINTS TO BUFFER
004421  104172                         MOV     (R0),R2         ; R2 HAS PATTERN NUMBER (IN EACH NIBBLE)
004422  103202  177760                 BIC     #LBLONB,R2      ; CLEAR UNUSED BITS
004424  054431                         BNE     1$              ; IF NOT PATTERN 16, BRANCH
004425  104200  000020  002221         MOV     #16.,PNUM       ; REPORT PATTERN 16
004430  004433                         BR      2$              ; BRANCH
004431  104020  002221         1$:     MOV     R2,PNUM         ; SAVE PATTERN NUMBER
004433  104021                 2$:     MOV     R2,R1           ; BUILD PATTERN NUMBER WORD IN R1
004434  110702                         SWAB    R2              ; MOVE PATTERN NUMBER TO HI BYTE
004435  101021                         BIS     R2,R1           ; SET HI BITS
004436  110202                         ROL     R2              ; ROTATE TO HI NIBBLE
004437  110202                         ROL     R2
004440  110202                         ROL     R2
004441  110202                         ROL     R2
004442  103202  007777                 BIC     #HBHINB,R2      ; CLEAR UNUSED BITS
004444  101021                         BIS     R2,R1           ; SET BITS
004445  110702                         SWAB    R2              ; MOVE TO LO BYTE
004446  101021                         BIS     R2,R1           ; SET BITS
004447  106271                         CMP     (R0)+,R1        ; SEE IF REDUNDANT PATTERN OK
004450  054514                         BNE     FWRD            ; IF NOT, BRANCH
004451  104302  002221                 MOV     PNUM,R2         ; RESTORE R2
004453  103202  177760                 BIC     #LBLONB,R2      ; MAP PATTERN 16 TO PATTERN 0
004455  104622  002236                 MOV     PATPTR(R2),R2   ; POINT TO PATTERN
004457  115402                         INC     R2              ; SKIP EDC
004460  104201  000001                 MOV     #1,R1           ; R1 HAS OFFSET INTO BUFFER
                                        .DSABL  LSB
004462  104024                 XOPLP0: MOV     R2,R4           ; R4 POINTS TO LENGTH OF PATTERN
004463  104243                         MOV     (R4)+,R3        ; R3 CONTAINS LENGTH OF PATTERN
004464  106203  000001                 CMP     #1,R3           ; SEE IF PATTERN IS 1 WORD LONG
004466  014501                         BEQ     ONEPAX          ; IF SO, BRANCH
004467  104245                 XOPLP1: MOV     (R4)+,R5        ; R5 GETS 1 WORD OF THE DATA PATTERN
004470  106275                         CMP     (R0)+,R5        ; COMPARE PATTERN WORD TO SECTOR AREA
004471  054515                         BNE     CMPERR          ; IF NOMATCH, BRANCH
004472  115401                         INC     R1              ; INCREMENT OFFSET
004473  106201  000400                 CMP     #SCTWRD+1,R1    ; SEE IF ENTIRE BUFFER COMPARED
004475  014510                         BEQ     CSCEXT          ; IF ALL WORDS COMPARED, BRANCH
004476  117403                         DEC     R3              ; DECREMENT COUNT OF WORDS IN PATTERN
004477  054467                         BNE     XOPLP1          ; IF DATA PATTERN UNFINISHED, BRANCH
```

```
     79 004500  004462                        BR      XOPLP0          ; BRANCH
     80 004501  104145                ONEPAX: MOV     (R4),R5         ; GET 1 WORD OF DATA PATTERN
     81 004502  106275                XOPLP2: CMP     (R0)+,R5        ; COMPARE PATTERN TO SECTOR READ
     82 004503  054515                        BNE     CMPERR          ; IF COMPARE ERROR, BRANCH
     83 004504  115401                        INC     R1              ; INCREMENT NUMBER OF WORDS TO COMPARE
     84 004505  106201  000400                CMP     #SCTWRD+1,R1    ; SEE IF ALL WORDS COMPARED
     85 004507  054502                        BNE     XOPLP2          ; IF INCOMPLETE, BRANCH
     86 004510  114002                CSCEXT: CLR     R2              ; NO ERRORS
     87 004511                                POP     <R5,R4>         ; RESTORE R5,R4
        004511  104265                                                                  MOV (SP)+,R5
        004512  104264                                                                  MOV (SP)+,R4
     88 004513  004707                        BR      CMXEX           ; BRANCH
     89 004514  114001                FWRD:   CLR     R1              ; ZERO OFFSET
     90 004515                        CMPERR: POP     <R5,R4>         ; RESTORE R5,R4
        004515  104265                                                                  MOV (SP)+,R5
        004516  104264                                                                  MOV (SP)+,R4
     91 004517                                PUSH    R0              ; SAVE POINTER TO BUFFER
        004517  100467                                                                  MOV R0,-(SP)
     92 004520  104037                        MOV     R3,R0           ; MOVE POINTER TO READ CHAIN TO R0
     93 004521  021725                        CALL    BLKCHK          ; SEE IF THIS IS A KNOWN BAD BLOCK
     94 004522                                POP     R0              ; RESTORE R0
        004522  104267                                                                  MOV (SP)+,R0
     95 004523  044707                        BCC     CMXEX           ; IF KNOWN BAD BLOCK, EXIT
     96 004524  104303  002232                MOV     CHAINS,R3       ; R3 POINTS TO LINK
     97 004526                                HARDER  12
        004526  104200  001113  001107                                         MOV     #ER12,HRQ.04
        004531  104202  107654                                                  MOV     #12!ERHARD+4000.,R2
        004533  104020  001105                                                  MOV     R2,HRQ.02
        004535  104200  004535  001104                                         MOV     #.,HRQ.01
        004540  104200  060014  001103                                         MOV     #ERRMC,HRQ.RQ
     98 004543                          CERROR  5,#SER24        ; ECC/EDC HAD DETECTED ERROR
        004543  104200  001326  001110                                         MOV     #SER24,HRQ.05
     99 004546  115000  002220          TST     SCR2            ; SEE IF ERROR DETECTED
    100 004550  054554                  BNE     4$              ; IF SO, BRANCH
    101 004551                          CERROR  5,#SER25        ; ECC/EDC DID NOT DETECT ERROR
        004551  104200  001354  001110                                         MOV     #SER25,HRQ.05
    102 004554                  4$:     ERRORC  <S.LETR(R4),RW.LOW(R3),RW.HI(R3)>
        004554  104640  000005  001111                                         MOV     S.LETR(R4),HRQ.06
        004557  104630  000002  001112                                         MOV     RW.LOW(R3),HRQ.07
        004562  104630  000003  001113                                         MOV     RW.HI(R3),HRQ.08
    103 004565                          ERRORC  <U.PARM(R5),#RBNTXT>
        004565  104650  000046  001114                                         MOV     U.PARM(R5),HRQ.09
        004570  104200  005472  001115                                         MOV     #RBNTXT,HRQ.10
    104 004573                          ERRORC  <U.RBN(R5),U.RBN+1(R5),RW.ANG(R3)>
        004573  104650  000055  001116                                         MOV     U.RBN(R5),HRQ.11
        004576  104650  000056  001117                                         MOV     U.RBN+1(R5),HRQ.12
        004601  104630  000006  001120                                         MOV     RW.ANG(R3),HRQ.13
    105 004604                          ERRORC  <RW.CMD(R3),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
        004604  104630  000004  001121                                         MOV     RW.CMD(R3),HRQ.14
        004607  104650  000066  001122                                         MOV     U.CGRP(R5),HRQ.15
        004612  104650  000064  001123                                         MOV     U.CCYL(R5),HRQ.16
        004615  104650  000065  001124                                         MOV     U.CCYL+1(R5),HRQ.17
    106 004620  106201  000366          CMP     #246.,R1        ; SEE IF IN LAST 9 WORDS
    107 004622  034633                  BPL     1$              ; IF NOT, BRANCH
    108 004623  104013                  MOV     R1,R3           ; TO COMPUTE OFFSET OF ERROR
    109 004624  107203  000367          SUB     #247.,R3        ; R3 HAS UNADJUSTED ERROR POSITION
    110 004626  105203  000003          ADD     #3,R3           ; R3 HAS ADJUSTED ERROR POSITION
```

```
    111 004630  107037                        SUB     R3,R0           ; ADJUST POINTER
    112 004631  117407                        DEC     R0              ; COMPARE AUTO-INCREMENTED, SO ADJUST
    113 004632  004646                        BR      3$              ; PRINT ERROR
    114 004633  106201  000004          1$:   CMP     #4,R1           ; SEE IF IN FIRST 4 WORDS
    115 004635  044643                        BCC     2$              ; IF SO, BRANCH
    116 004636  107207  000004                SUB     #4,R0           ; LOOK BACK FOUR WORDS
    117 004640  104203  000003                MOV     #3,R3           ; OFFSET OF ERROR IN PRINTOUT IS 3
    118 004642  004646                        BR      3$              ; BRANCH
    119 004643  107017                  2$:   SUB     R1,R0           ; BACK UP POINTER
    120 004644  104013                        MOV     R1,R3           ; R3 IS OFFSET OF ERROR IN PRINTOUT
    121 004645  117407                        DEC     R0              ; TO POINT TO START OF BUFFER
    122 004646                          3$:   ERRORC  <PNUM,R1,R3,(R0)+,(R0)+,(R0)+,(R0)+,(R0)+,(R0)+,(R0)+,(R0)+>
        004646  104300  002221  001125                                        MOV     PNUM,HRQ.18
        004651  104010  001126                                              MOV     R1,HRQ.19
        004653  104030  001127                                              MOV     R3,HRQ.20
        004655  104270  001130                                              MOV     (R0)+,HRQ.21
        004657  104270  001131                                              MOV     (R0)+,HRQ.22
        004661  104270  001132                                              MOV     (R0)+,HRQ.23
        004663  104270  001133                                              MOV     (R0)+,HRQ.24
        004665  104270  001134                                              MOV     (R0)+,HRQ.25
        004667  104270  001135                                              MOV     (R0)+,HRQ.26
        004671  104270  001136                                              MOV     (R0)+,HRQ.27
        004673  104270  001137                                              MOV     (R0)+,HRQ.28
    123 004675                              ERRORC  <(R0)+,(R0)+,(R0)+,(R0)+>
        004675  104270  001140                                              MOV     (R0)+,HRQ.29
        004677  104270  001141                                              MOV     (R0)+,HRQ.30
        004701  104270  001142                                              MOV     (R0)+,HRQ.31
        004703  104270  001143                                              MOV     (R0)+,HRQ.32
    124 004705                              ENDERR  0
        004705  114000  002230                                              CLR     ERRPOS          ; CLEAR THE POSITION
    125 004707  104207  000021  CMXEX:     MOV     #LSTMOD,R0      ; LAST MODULE WILL BE CALLED
    126 004711  114001                      CLR     R1              ; CALL IMMIDATELY
    127 004712  003231                      BR      JMPRET          ; RETURN TO SEQUNCR
    131                                     .IF     LE,BUFARA-.
    132                             BUFARA  =       .+1
    133                                     .ENDC
    134                                     .IF     LE,MAXADR-.
    135                             MAXADR  =       .+1
    136                                     .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE LSTMOD - CLEANUP MODULE BEFORE SETUP
                                        DMOVLY  LM,AREA0
004713  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
000021                          LSTMOD  =       CMPDAT+1
                                ;
                                ;       LOOP CONTROL FOR ECC, EDC AND DATA COMPARE MODULES.  ALSO LAST
                                ;       MODULE BEFORE SETUP (USUALLY)
                                ;
                                        .ENABL  LSB
004413  114002                         CLR     R2              ; ASSUME NO ERRORS
004414  104657  000021                 MOV     U.NSEC(R5),R0   ; GET NUMBER OF SECTORS HANDLED SO FAR
004416  115407                         INC     R0              ; INCREMENT COUNT
004417  100657  000021                 MOV     R0,U.NSEC(R5)   ; SAVE
004421  104701  175607                 MOV     @CHAINS,R1      ; SEE IF LAST BUFFER LAST READ
004423                                  ASSUME  RW.STAT,0       ; ASSUME STATUS FIRST WORD
004423  074434                         BMI     1$              ; IF SO, BRANCH
004424                                  ASSUME  EOC,100000      ; ASSUME END-OF-CHAIN SIGN BIT
004424  103201  170000                 BIC     #UNADDR,R1      ; CLEAR UNUSED ADDRESSING BITS
004426  104010  002232                 MOV     R1,CHAINS       ; CHAINS NOW POINTS TO NEXT LINK
004430  104207  000013                 MOV     #SECCHK,R0      ; SECTOR CHECK NEXT MODULE
004432  114001                         CLR     R1              ; IMMIDATE CALL
004433  004467                         BR      4$              ; EXIT
004434  104653  000046         1$:     MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
004436  103203  000200                 BIC     #RBNBN,R3       ; IF HANDLING AN RBN, CLEAR IT
004440  100653  000046                 MOV     R3,U.PARM(R5)   ; SAVE
004442  105647  000010                 ADD     S.MEGR(R4),R0   ; GET MEGABYTE COUNT
004444  106207  003642                 CMP     #1[illegible],R0 ; SEE IF ONE MEGABYTE TRANSFERED
004446  034462                         BPL     3$              ; IF NOT, BRANCH
004447  107207  003642                 SUB     #1953.,R0       ; ZERO COUNT
004451  104200  000001  001105         MOV     #1,HRQ.02       ; REPORT 1 MEGABYTE READ
004454  114000  001106                 CLR     HRQ.03          ; NO WRITE MEGABYTES REPORTED
004456  104202  060011                 MOV     #T4MXFR,R2      ; SET UP FOR MEGABIT REPORT
004460  104020  001103                 MOV     R2,HRQ.RQ       ; FLAG AS NON-ERROR
004462  100647  000010         3$:     MOV     R0,S.MEGR(R4)   ; SAVE COUNT
004464  104207  000001                 MOV     #SETUP,R0       ; SETUP IS NEXT MODULE CALLED
004466  104051                         MOV     R5,R1           ; DEFERRED CALL TO NEXT MODULE
004467  003231                 4$:     BR      JMPRET          ; RETURN TO SEQUENCER
                                        .DSABL  LSB
                                        .IF     LE,BUFARA-.
                                BUFARA  =       .+1
                                        .ENDC
                                        .IF     LE,MAXADR-.
                                MAXADR  =       .+1
                                        .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE REVCT - REVECTORED SECTOR HANDLING
004470                                  DMOVLY  RV,AREA0
004470  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000022                          REVCT   =       LSTMOD+1        ; REVECTOR OVERLAY
                                ;       REVECTOR A SECTOR WITH A HEADER NOT FOUND OR A SECONDARY REVECTOR
                                        .ENABL  LSB
004413  104657  000046                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
004415  102207  000400                  BIT     #REVEC,R0       ; SEE IF REVECTOR ALLREADY IN PROGRESS
004417  054475                          BNE     1$              ; IF SO, BRANCH
                                .SBTTL  REVSUP - SETUP THE REVECTOR OPERATION (TO READ RCT)
                                ;REVSUP
                                ;
                                ;       INITILIZE ALL PARAMETERS FOR READING THE RCT TO FING THE RBN THAT
                                ;       A HEADER HAS BEEN REVECTORED TO
                                ;
004420  101207  000400                  BIS     #REVEC,R0       ; FLAG AS REVECTOR IN PROGRESS
004422  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE PARAMETERS
004424  104207  005523                  MOV     #RCTL$,R0       ; POINT OT RCT LBN STRING
004426  100647  000005                  MOV     R0,S.LETR(R4)   ; SAVE
004430  104657  000061                  MOV     U.RWER(R5),R0   ; GET READ/WRITE ERROR TYPE
004432  100657  000062                  MOV     R0,U.RVER(R5)   ; SAVE FOR REVECTOR INFORMATION
004434  104641  000007                  MOV     S.SCHR(R4),R1   ; R1 POINTS TO SUBUNIT CHARACTERISTICS
004436  114007                          CLR     R0              ; USE R0 TO INITILIZE VALUES
004437  100657  000055                  MOV     R0,U.RBN(R5)    ; START WITH RBN ZERO
004441  100657  000056                  MOV     R0,U.RBN+1(R5)  ; START WITH RBN ZERO
004443  100657  000060                  MOV     R0,U.CCOP(R5)   ; ON ORIGINAL COPY OF RCT
004445  115407                          INC     R0              ; R0 IS NOW 1
004446  100657  000022                  MOV     R0,U.MSEC(R5)   ; ONLY READ 1 SECTOR AT A TIME
004450  115407                          INC     R0              ; R0 IS NOW 2
004451  105617  000012                  ADD     LBNHST(R1),R0   ; R0 POINTS TO 1ST REVECTOR INFORMATION SECTOR
004453  100657  000053                  MOV     R0,U.CBN(R5)    ; SAVE
004455  104617  000013                  MOV     LBNHST+1(R1),R0 ; R0 HAS HI FIRST REV INFO SECTOR
004457  044461                          BCC     5$              ; IF NO CARY, BRANCH
004460  115407                          INC     R0              ; PROPOGATE CARRY
004461  103207  170000          5$:     BIC     #^CHBHINB,R0    ; CLEAR SUBUNIT BITS
004463  100657  000054                  MOV     R0,U.CBN+1(R5)  ; SAVE
004465  114002                          CLR     R2              ; NO ERRORS
004466  104201  177777                  MOV     #-1,R1          ; START READ ATTEMPT RETRIES AT 0
004470  100651  000012                  MOV     R1,U.RWTO(R5)   ; SAVE
004472  104207  000023                  MOV     #SEEK,R0        ; SEEK IS NEXT MODULE
004474  004524                          BR      4$              ; EXIT
004475  104651  000024          1$:     MOV     U.CSEC(R5),R1   ; GET NUMBER OF SECTORS R/W SO FAR
004477  105651  000051                  ADD     U.MBN(R5),R1    ; ADD STARTING SECTOR OF OPERATION
004501  104010  002175                  MOV     R1,CURBN        ; MOVE TO TEMP STORAGE
004503  104651  000052                  MOV     U.MBN+1(R5),R1  ; GET HI STARTING SECTOR
004505  044507                          BCC     2$              ; IF NO CARRY, BRANCH
004506  115401                          INC     R1              ; PROPOGATE CARRY
004507  104010  002176          2$:     MOV     R1,CURBN+1      ; SAVE
004511  104301  002220                  MOV     SCR2,R1         ; SEE IF SECTOR READ IS OK
004513  014516                          BEQ     6$              ; IF SO, BRANCH
```

REVSUP - SETUP THE REVECTOR OPERATION (TO READ RCT)

```
     81 004514  024525                        CALL    REVSOK          ; FIND NEXT COPY TO READ
     82 004515  004524                        BR      4$              ; EXIT
     83 004516  024635              6$:       CALL    SEARCH          ; SEARCH THE SECTOR TO FIND THE LBN
     84 004517  115001                        TST     R1              ; SEE IF LBN FOUND
     85 004520  014524                        BEQ     4$              ; LBN FOUND, BRANCH AND READ RBN
     86 004521  115002                        TST     R2              ; SEE IF NULL FLAG FOUND (REVECTOR NOT FOUND)
     87 004522  054524                        BNE     4$              ; IF NOT, BRANCH
     88 004523  024774                        CALL    NXTRCT          ; READ NEXT RCT SECTOR
     89 004524  003231              4$:       BR      JMPRET          ; RETURN CONTROL TO SEQUENCER
     90                                       .DSABL  LSB
```

```
                                        .SBTTL  REVSOK - SEE IF THE REVECTOR INFO SECTOR JUST READ IS OK
004525                                  REVSOK:
                                        ;
                                        ;       GO TO THE NEXT RCT COPY
                                        ;
004525  104651  000060                  MOV     U.CCOP(R5),R1   ; GET NUMBER OF COPIES TRIED SO FAR
004527  115401                          INC     R1              ; TRY ANOTHER COPY
004530  106651  000057                  CMP     U.COPY(R5),R1   ; CHECK AGAINST MAX
004532  034604                          BPL     5$              ; IF ALL COPIES UNTRIED, BRANCH
004533                                  HARDER  40
004533  104200  002705  001107                                          MOV     #ER40,HRQ.04
004536  104202  107710                                                  MOV     #40!ERHARD+4000.,R2
004540  104020  001105                                                  MOV     R2,HRQ.02
004542  104200  004542  001104                                          MOV     #.,HRQ.01
004545  104200  060014  001103                                          MOV     #ERRMC,HRQ.RQ
004550  104651  000062                  MOV     U.RVER(R5),R1   ; GET ORIGINAL ERROR TYPE
004552  106201  000003                  CMP     #3,R1           ; SEE IF REVECTORED BLOCK
004554  054561                          BNE     2$              ; IF NOT, BRANCH
004555                                  CERROR  5,#SER26        ; FLAG ERROR
004555  104200  003035  001110                                          MOV     #SER26,HRQ.05
004560  004564                          BR      36$             ; EXIT
004561                          2$:     CERROR  5,#SER32        ; FLAG ERROR HEADER COMPARE ERROR
004561  104200  003051  001110                                          MOV     #SER32,HRQ.05
004564                          36$:    ERRORC  <U.CBN(R5),U.CBN+1(R5),CURBN,CURBN+1>
004564  104650  000053  001111                                          MOV     U.CBN(R5),HRQ.06
004567  104650  000054  001112                                          MOV     U.CBN+1(R5),HRQ.07
004572  104300  002175  001113                                          MOV     CURBN,HRQ.08
004575  104300  002176  001114                                          MOV     CURBN+1,HRQ.09
004600                                  ENDERR  0
004600  114000  002230                                                  CLR     ERRPOS          ; CLEAR THE POSITION
004602  025125                          CALL    RVFAIL          ; CLEAR ALL BITS AND DO A FAIL EXIT
004603  004634                          BR      7$              ; EXIT
004604  100651  000060          5$:     MOV     R1,U.CCOP(R5)   ; SAVE COPY COUNT
004606  104647  000007                  MOV     S.SCHR(R4),R0   ; R0 POINTS TO SUBUNIT PARAMETERS
004610  104677  000014                  MOV     RCTCSZ(R0),R0   ; R0 IS RCT COPY SIZE
004612  105657  000053                  ADD     U.CBN(R5),R0    ; ADD CURRENT SECTOR
004614  100657  000053                  MOV     R0,U.CBN(R5)    ; SAVE
004616  044624                          BCC     6$              ; IF NO CARRY, BRANCH
004617  104657  000054                  MOV     U.CBN+1(R5),R0  ; GET HI CURRENT BLOCK NUMBER
004621  115407                          INC     R0              ; INCREMENT COUNT
004622  100657  000054                  MOV     R0,U.CBN+1(R5)  ; SAVE
004624  024163                  6$:     CALL    ENABLE
004625  104203  177777                  MOV     #-1,R3          ; START READ ATTEMPT RETRIES AT 0
004627  100653  000012                  MOV     R3,U.RWTO(R5)   ; SAVE
004631  104207  000023                  MOV     #SEEK,R0        ; SEEK IS NEXT MODULE
004633  114002                          CLR     R2              ; NO ERRORS
004634  000000                  7$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL   EARCH - TRY TO FIND THE LBN IN THE RCT SECTOR JUST READ
004635                          SEARCH:
                                ;
                                ;       SEARCH THE RCT SECTOR JUST READ TO FIND THE LBN OR THE NULL ENTRY
                                ;
004635  104307  002232                 MOV     CHAINS,R0       ; R0 POINTS TO LINK (NODE) IN READ CHAIN
004637  104677  000001                 MOV     RW.BUF(R0),R0   ; R0 NOW POINTS TO BUFFER
004641  114001                         CLR     R1              ; R1 IS RBN NUMBER OFFSET WITHIN THE SECTOR
004642  104672  000001         1$:     MOV     1(R0),R2        ; GET RCT CODE
004644  074730                         BMI     6$              ; IF NULL POINTER, ENTIRE TABLE EXHAUSTED. BRANCH
004645  102202  020000                 BIT     #020000,R2      ; SEE IF IT IS A USED RBN
004647  014706                         BEQ     5$              ; IF NOT, BRANCH
004650  103202  170000                 BIC     #^CHBHINB,R2    ; CLEAR CODE
004652  106302  002176                 CMP     CURBN+1,R2      ; SEE IF HI ORDER MATCHES
004654  054706                         BNE     5$              ; IF NOT, BRANCH
004655  106170  002175                 CMP     (R0),CURBN      ; SEE IF LO ORDER MATCHES
004657  054706                         BNE     5$              ; IF NOT, BRANCH
004660  105651  000055                 ADD     U.RBN(R5),R1    ; ADD RUNNING RBN TO OFFSET
004662  100651  000055                 MOV     R1,U.RBN(R5)    ; SAVE
004664  044672                         BCC     2$              ; IF NO CARRY, BRANCH
004665  104651  000056                 MOV     U.RBN+1(R5),R1  ; GET HI ORDER RBN
004667  115401                         INC     R1              ; PROPOGATE CARRY
004670  100651  000056                 MOV     R1,U.RBN+1(R5)  ; SAVE
004672  104657  000046         2$:     MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
004674  101207  000200                 BIS     #RBNBN,R0       ; FLAG ALL ROUTINES THAT THIS IS AN RBN
004676  100657  000046                 MOV     R0,U.PARM(R5)   ; SAVE
004700  025125                         CALL    RVFAIL          ; FAIL EXIT DOES WHAT WE WANT, JUST CLEAR R1 AND R2
004701  114001                         CLR     R1              ; FOUND IT
004702  114002                         CLR     R2              ; NO ERRORS
004703  100652  000021                 MOV     R2,U.NSEC(R5)   ; SAVE
004705  004773                         BR      7$              ; EXIT
004706  105207  000002         5$:     ADD     #2,R0           ; POINT TO NEXT RBN RECORD
004710  115401                         INC     R1              ; INCREMENT RBN OFFSET
004711  106201  000200                 CMP     #128.,R1        ; SEE IF ALL RECORDS TRIED
004713  054642                         BNE     1$              ; IF NOT, BRANCH
004714  114002                         CLR     R2              ; TO SIGNAL CALLING ROUTINE TO READ NEXT SECTOR
004715  105651  000055                 ADD     U.RBN(R5),R1    ; ADD OLD RBN TO 128
004717  100651  000055                 MOV     R1,U.RBN(R5)    ; SAVE
004721  044773                         BCC     7$              ; IF NO CARRY, EXIT
004722  104651  000056                 MOV     U.RBN+1(R5),R1  ; GET HI ORDER RBN
004724  115401                         INC     R1              ; PROPOGATE CARRY
004725  100651  000056                 MOV     R1,U.RBN+1(R5)  ; SAVE
004727  004773                         BR      7$              ; EXIT
004730                          6$:     HARDER  41
004730  104200  002775  001107                                                 MOV     #ER41,HRQ.04
004733  104202  107711                                                         MOV     #41!ERHARD+4000.,R2
004735  104020  001105                                                         MOV     R2,HRQ.02
004737  104200  004737  001104                                                 MOV     #.,HRQ.01
004742  104200  060014  001103                                                 MOV     #ERRMC,HRQ.RQ
004745  104651  000062                 MOV     U.RVER(R5),R1   ; GET ORIGINAL ERROR
004747  106201  000003                 CMP     #3,R1           ; SEE IF REVECTORED BLOCK
004751  054756                         BNE     32$             ; IF NOT, BRANCH
004752                                 CERROR  5,#SER26        ; REVECTORED SECTOR
004752  104200  003035  001110                                                 MOV     #SER26,HRQ.05
004755  004761                         BR      12$             ; EXIT
004756                          32$:    CERROR  5,#SER32        ; REPORT HEADER COMPARE ERROR
004756  104200  003051  001110                                                 MOV     #SER32,HRQ.05
```

SEARCH - TRY TO FIND THE LBN IN THE RCT SECTOR JUST READ

```
     51 004761                          12$:    ERRORC  <CURBN,CURBN+1>
        004761  104300  002175  001111                                    MOV     CURBN,HRQ.06
        004764  104300  002176  001112                                    MOV     CURBN+1,HRQ.07
     52 004767                                  ENDERR  0
        004767  114000  002230                                            CLR     ERRPOS          ; CLEAR THE POSITION
     53 004771  025125                  9$:     CALL    RVFAIL          ; CLEAR ALL BITS, FAIL EXIT
     54 004772  104051                          MOV     R5,R1           ; FLAG AS ERROR
     55 004773  000000                  7$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  NXTRCT - GET THE NEXT RCT SECTOR TO SEARCH
004774                          NXTRCT:
                                ;
                                ;       INCREMENT CBN BY 1 AFTER SUBTACTING THE NUNMBER OF COPIES*RCT SIZE
                                ;       THAT WAS SEARCHED THROUGH ON THE LAST PASS
                                ;
004774  104652  000053                MOV     U.CBN(R5),R2    ; R2 HAS LO ORDER RCT BN
004776  104653  000054                MOV     U.CBN+1(R5),R3  ; R3 HAS HI ORDER RCT BN
005000  104657  000060                MOV     U.CCOP(R5),R0   ; GET NUMBER OF COPIES THAT HAVE BEEN READ
005002  015013                        BEQ     3$              ; IF NO COPIES, BRANCH
005003  104641  000007                MOV     S.SCHR(R4),R1   ; R1 POINTS TO SUBUNIT CHARACTERISTICS
005005  107612  000014        1$:     SUB     RCTCSZ(R1),R2   ; SUBTRACT FROM LO ORDER BN
005007  045011                        BCC     2$              ; IF NO BORROW, BRANCH
005010  117403                        DEC     R3              ; PROPOGATE BORROW
005011  117407                2$:     DEC     R0              ; DECREMENT COUNT
005012  055005                        BNE     1$              ; IF NO CARRY, BRANCH
005013  105202  000001        3$:     ADD     #1,R2           ; INCREMENT CBN BY 1
005015  045017                        BCC     4$              ; IF NO CARRY, BRANCH
005016  115403                        INC     R3              ; PROPOGATE CARRY
005017  100652  000053        4$:     MOV     R2,U.CBN(R5)    ; SAVE LO ORDER
005021  100653  000054                MOV     R3,U.CBN+1(R5)  ; SAVE HI ORDER
005023  104020  002175                MOV     R2,CURBN        ; MOVE TO CALC AREA
005025  104030  002176                MOV     R3,CURBN+1      ; MOVE TO CALC AREA
                                ;                               NOTE R0 MUST BE ZERO FOR CALC SETUP
005027  022027                        CALL    CALC            ; FIND CYL THAT NEXT SEC IS ON
005030  104647  000007                MOV     S.SCHR(R4),R0   ; POINT TO SUBUNIT CHARACTERISTICS
005032  106670  000001  002202        CMP     HICYL(R0),CYL+1 ; SEE IF IN XBN AREA
005035                                BCS     5$              ; IF SO, BRANCH
005035  045037                                                                BCC     .+2
005036  005045                                                                BR      5$
005037  055114                        BNE     6$              ; IF NOT, BRANCH
005040  106670  000000  002201        CMP     LBNCYL(R0),CYL  ; SEE IF IN XBN AREA
005043  015045                        BEQ     5$              ; IF SO, BRANCH
005044  045114                        BCC     6$              ; IF NOT, BRANCH
005045  025125                5$:     CALL    RVFAIL          ; REVECTOR FAILED
005046                                HARDER  4               ; REPORT CORRUPTED RCT
005046  104200  000271  001107                                        MOV     #ER4,HRQ.04
005051  104202  107644                                                MOV     #4!ERHARD+4000.,R2
005053  104020  001105                                                MOV     R2,HRQ.02
005055  104200  005055  001104                                        MOV     #.,HRQ.01
005060  104200  060014  001103                                        MOV     #ERRMC,HRQ.RQ
005063  104651  000062                MOV     U.RVER(R5),R1   ; GET ORIGINAL ERROR
005065  106201  000003                CMP     #3,R1           ; SEE IF REVECTOR
005067  055074                        BNE     7$              ; IF NOT, BRANCH
005070                                CERROR  5,#SER26        ; REPORT REVECTOR
005070  104200  003035  001110                                        MOV     #SER26,HRQ.05
005073  005077                        BR      8$              ; BRANCH
005074                          7$:     CERROR  5,#SER32        ; REPORT HEADER COMPARE ERROR
005074  104200  003051  001110                                        MOV     #SER32,HRQ.05
005077                          8$:     ERRORC  <CURBN,CURBN+1,U.CBN(R5),U.CBN+1(R5)>
005077  104300  002175  001111                                        MOV     CURBN,HRQ.06
005102  104300  002176  001112                                        MOV     CURBN+1,HRQ.07
005105  104650  000053  001113                                        MOV     U.CBN(R5),HRQ.08
005110  104650  000054  001114                                        MOV     U.CBN+1(R5),HRQ.09
005113  005124                        BR      9$              ; REPORT ERROR
005114  114002                6$:     CLR     R2              ; NO ERRORS
005115  024163                        CALL    ENABLE          ; ENABLE ERROR RECOVERY
```

NXTRCT - GET THE NEXT RCT SECTOR TO SEARCH

```
     45 005116  104203  177777                MOV     #-1,R3          ; START READ ATTEMPT RETRIES AT 0
     46 005120  100653  000012                MOV     R3,U.RWTO(R5)   ; SAVE
     47 005122  104207  000023                MOV     #SEEK,R0        ; NEXT MODULE IS SEEK
     48 005124  000000                  9$:   RETURN
```

```
                                        .SBTTL  RVFAIL - CLEAR ALL REVECTOR BITS, AND CALL SETUP NEXT
005125                                  RVFAIL:
                                        ;
                                        ;       CLEAR ALL REVECTOR BITS, AND RESTORE U.CBN TO WHAT IT WAS
                                        ;
005125  104651  000046                          MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
005127  103201  000400                          BIC     #REVEC,R1       ; CLEAR ALL REVECTOR BITS
005131  100651  000046                          MOV     R1,U.PARM(R5)   ; SAVE
005133  104201  000001                          MOV     #1,R1           ; FLAG THAT A SECTOR HAS BEEN READ (SKIPPED)
005135  100651  000021                          MOV     R1,U.NSEC(R5)   ; SAVE
005137  104651  000024                          MOV     U.CSEC(R5),R1   ; GET NUMBER OF SECTORS
005141  105651  000051                          ADD     U.MBN(R5),R1    ; ADD STARTING SECTOR NUMBER
005143  100651  000053                          MOV     R1,U.CBN(R5)    ; RESTORE CBN
005145  104651  000052                          MOV     U.MBN+1(R5),R1  ; GET HI MBN
005147  045151                                  BCC     1$              ; IF NO CARRY, BRANCH
005150  115401                                  INC     R1              ; PROPOGATE CARRY
005151  100651  000054                1$:       MOV     R1,U.CBN+1(R5)  ; SAVE HI IN CBN
005153  104207  005517                          MOV     #L$,R0          ; GET LBN STRING
005155  100647  000005                          MOV     R0,S.LETR(R4)   ; SAVE
005157  104207  000001                          MOV     #SETUP,R0       ; SETUP NEXT ROUTINE
005161  000000                                  RETURN                  ; RETURN TO CALLING PROGRAM
                                        .IF     LE,BUFARA-.
                                BUFARA  =       .+1
                                        .ENDC
                                        .IF     LE,MAXADR-.
                                MAXADR  =       .+1
                                        .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE SEEK - IF NECESSARY, ISSUE SEEK
                                        DMOVLY  SK,BUFARA
005162  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
000023                          SEEK    =       REVCT+1         ; SEEK OVERLAY
                                ;
                                ;       SEEK ISSUES A SEEK TO THE CYLINDER SPECIFIED IN U.CCYL(R5) (2 WORDS)
                                ;       AND GROUP GIVEN IN U.CGRP(R5) (1 WORD).
                                ;       IT THEN RETURNS WITH A DEFERED CALL TO SEKTST (TEST SEEK) TO SEE IF
                                ;       THE SEEK COMPLETED SUCCESSFULLY.  IF SO, THE READ WRITE MODULE IS
                                ;       THEN CALLED
                                ;
                                        .ENABL  LSB
005230  104653  000046                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
005232  102203  002000                  BIT     #SEKINP,R3      ; SEE IF SEEK IS ALLREADY IN PROGRESS
005234  055241                          BNE     3$              ; IF SO, BRANCH
005235  104201  177777                  MOV     #-1,R1          ; FOR ZEROING FOLLOWING WORDS
005237  100651  000007                  MOV     R1,U.SRTY(R5)   ; SAVE
005241  104653  000047          3$:     MOV     U.RCOV(R5),R3   ; GET RECOVERY WORD
005243  102203  002000                  BIT     #RCBREQ,R3      ; SEE IF RECALIBRATION REQUESTED
005245  015252                          BEQ     4$              ; IF NOT, BRANCH
005246  104207  000025                  MOV     #RECALB,R0      ; RECALIBRATE NEXT MODULE
005250  114002                          CLR     R2              ; NO ERRORS
005251  005422                          BR      SEKOUT          ; EXIT
005252  104652  000007          4$:     MOV     U.SRTY(R5),R2   ; GET SEEK RETRIES
005254  115402                          INC     R2              ; ADJUST FOR TEST
005255  015371                          BEQ     5$              ; IF FIRST TIME, BRANCH
005256  106202  000002                  CMP     #2,R2           ; SEE IF TRIED MAX NUM OF TIMES
005260  045347                          BCC     1$              ; IF NOT, BRANCH
005261                                  DEVFTL  14,<S.LETR(R4),U.CBN(R5),U.CBN+1(R5)>
005261  104200  001435  001107                                          MOV     #ER14,HRQ.04
005264  104640  000005  001110                                          MOV     S.LETR(R4),HRQ.05
005267  104650  000053  001111                                          MOV     U.CBN(R5),HRQ.06
005272  104650  000054  001112                                          MOV     U.CBN+1(R5),HRQ.07
005275  104202  047656                                                  MOV     #14!FTLDEV+4000.,R2
005277  104020  001105                                                  MOV     R2,HRQ.02
005301  104200  005301  001104                                          MOV     #.,HRQ.01
005304  104200  060014  001103                                          MOV     #ERRMC,HRQ.RQ
005307                                  ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5),U.RBN+1(R5),GROUP,CYL,CYL+1>
005307  104650  000046  001113                                          MOV     U.PARM(R5),HRQ.08
005312  104200  005472  001114                                          MOV     #RBNTXT,HRQ.09
005315  104650  000055  001115                                          MOV     U.RBN(R5),HRQ.10
005320  104650  000056  001116                                          MOV     U.RBN+1(R5),HRQ.11
005323  104300  002203  001117                                          MOV     GROUP,HRQ.12
005326  104300  002201  001120                                          MOV     CYL,HRQ.13
005331  104300  002202  001121                                          MOV     CYL+1,HRQ.14
005334                                  ENDERR  0
005334  114000  002230                                                  CLR     ERRPOS          ; CLEAR THE POSITION
005336  104653  000046                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
005340  103203  002000                  BIC     #SEKINP,R3      ; CLEAR SEEK IN PROGRESS BIT
005342  100653  000046                  MOV     R3,U.PARM(R5)   ; SAVE
```

```
********** OVERLAY MODULE SEEK - IF NECESSARY, ISSUE SEEK

     005344  024215                          CALL    GORCLB          ; RECALIBRATE DRIVE
     005345  104051                          MOV     R5,R1           ; DEFFERED CALL
     005346  005423                          BR      SEKEXT          ; EXIT
     005347                          1$:     REPSFT  ,,SEEK          ; REPORT SEEK ERROR
     005347  114000  001105                                                  CLR     HRQ.02
     005351  114000  001106                                                  CLR     HRQ.03
     005353  104200  000001  001107                                          MOV     #1,HRQ.04
     005356  100467                                                                  MOV R0,-(SP)
     005357  104657  000063                                                  MOV     U.UNUM(R5),R0
     005361  105657  000050                                                  ADD     U.SUBU(R5),R0
     005363  104070  001104                                                  MOV     R0,HRQ.01
     005365  104207  060007                                                  MOV     #T4SOFT,R0
     005367  021053                                                          CALL    HOSTRQ
     005370  104267                                                                  MOV (SP)+,R0
     005371  025424                  5$:     CALL    TSTNEC          ; SEE IF SEEK IS NECESSARY
     005372  115003                          TST     R3              ; SEE IF SEEK IS NECESSARY
     005373  015417                          BEQ     NOSEEK          ; IF ZERO (NO SEEK NECESSARY), BRANCH
     005374  025473                          CALL    ISUSEK          ; ISSUE SEEK
     005375  115002                          TST     R2              ; SEE IF ERROR OCCURRED
     005376  055423                          BNE     SEKEXT          ; IF ERROR, BRANCH
     005377  104651  000007                  MOV     U.SRTY(R5),R1   ; GET SEEK TIMEOUT
     005401  115401                          INC     R1              ; INCREMENT COUNT
     005402  100651  000007                  MOV     R1,U.SRTY(R5)   ; SAVE
     005404  104657  000011                  MOV     U.MSTO(R5),R0   ; MOVE TIMEOUT TO PARAMETERS
     005406  100657  000005                  MOV     R0,U.TIMH(R5)   ; INITILIZE TIMEOUT VALUE
                                     ;       NOTE:   R2 IS ZERO HERE
     005410  100652  000006                  MOV     R2,U.TIML(R5)   ; SAVE
     005412  104051                          MOV     R5,R1           ; DEFERRED CALL TO SEKTST
     005413  104207  000024                  MOV     #SEKTST,R0      ; SEKTST NEXT TO CALL
     005415  024172                          CALL    DSABLE          ; DISABLE ERROR RECOVERY DURING SEEK
     005416  005423                          BR      SEKEXT          ; BRANCH
     005417  104207  000007          NOSEEK: MOV     #BUILDP,R0      ; READ/WRITE ROUTINE NEXT
     005421  114002                          CLR     R2              ; NO ERRORS
     005422  114001                  SEKOUT: CLR     R1              ; IMMIDATE CALL
     005423  003231                  SEKEXT: BR      JMPRET          ; RETURN TO CALLING PROGRAM
                                             .DSABL  LSB
```

```
                                        .SBTTL  TSTNEC - TEST TO SEE IF SEEK IS NECESSARY
005424                                  TSTNEC:
                                        ;
                                        ;       TSTNEC TESTS TO SEE IF A SEEK IS NECESSARY.  IF NOT, R3 IS RETURNED
                                        ;       AS ZERO, NONZERO OTHERWISE.
                                        ;
005424  104657  000046                          MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005426  102207  000200                          BIT     #RBNBN,R0       ; SEE IF BLOCK REVECTORED
005430  015440                                  BEQ     2$              ; IF NOT, BRANCH
005431  104650  000055  002175                  MOV     U.RBN(R5),CURBN ; MOVE RBN TO CALCULATION AREA
005434  104650  000056  002176                  MOV     U.RBN+1(R5),CURBN+1     ; MOVE RBN TO CALCULATION AREA
005437  005446                                  BR      3$              ; BRANCH
005440  104650  000053  002175  2$:             MOV     U.CBN(R5),CURBN ; MOVE LBN TO CALCULATION AREA
005443  104650  000054  002176                  MOV     U.CBN+1(R5),CURBN+1     ; MOVE LBN TO CALCULATION AREA
005446  114007                  3$:             CLR     R0              ; TELL CALC ROUTINE TO SET UP PARAMETERS
005447  022027                                  CALL    CALC            ; CALCULATE CYL AND GROUP
005450  104203  002201                          MOV     #CYL,R3         ; R3 POINTS TO CALCULATED CYLINDER
005452  104657  000047                          MOV     U.RCOV(R5),R0   ; GET UNIT PARAMETERS
005454  102207  004000                          BIT     #SEKREQ,R0      ; SEE IF SEEK MANDATORY
005456  055472                                  BNE     9$              ; IF SO, BRANCH
005457  104202  000064                          MOV     #U.CCYL,R2      ; R2 WILL POINT TO CURRENT CYL
005461  105052                                  ADD     R5,R2           ; R2 POINTS TO CURRENT CYLINDER
005462  104201  000003                          MOV     #3,R1           ; GET COUNT
005464  104237                  4$:             MOV     (R3)+,R0        ; GET WORD
005465  106227                                  CMP     (R2)+,R0        ; SEE IF CYL AND GROUP THE SAME
005466  055472                                  BNE     9$              ; IF NOT, BRANCH
005467  117401                                  DEC     R1              ; DECREMENT COUNT
005470  055464                                  BNE     4$              ; IF COUNT INCOMPLETE, BRANCH
005471  114003                                  CLR     R3              ; FLAG AS SEEK NOT NECESSARY
005472  000000                  9$:             RETURN                  ; RETURN TO SEEK
```

```
                                        .SBTTL  ISUSEK - ISSUE SEEK COMMAND
005473                                  ISUSEK:
                                        ;
                                        ;       ISUSEK ISSUES THE SEEK TO THE DRIVE
                                        ;
005473  104300  002201  001673                 MOV     CYL,INS+1       ; MOVE LOW CYL TO SEEK COMMAND
005476  104300  002202  001674                 MOV     CYL+1,INS+2     ; SET LOWER BITS
005501  104300  002203  001675                 MOV     GROUP,INS+3     ; MOVE GROUP TO SEEK COMMAND
005504  104203  001636                         MOV     #CR.SEK,R3      ; SET UP FOR TALK
005506  021173                                 CALL    TALK            ; SEND SEEK
005507  115002                                 TST     R2              ; SEE IF ERROR OCCURRED
005510  055546                                 BNE     10$             ; IF SO, BRANCH
005511  104207  000064                         MOV     #U.CCYL,R0      ; R0 WILL POINT TO CURRENT CYLINDER
005513  105057                                 ADD     R5,R0           ; R0 POINTS TO CURRENT CYLINDER
005514  104201  000003                         MOV     #3,R1           ; MOVE THREE WORDS
005516                                         PUSH    <R0,R1>         ; SAVE POINTER AND COUNT
005516  100467                                                                 MOV R0,-(SP)
005517  100461                                                                 MOV R1,-(SP)
005520  104202  000067                         MOV     #U.LCYL,R2      ; R2 WILL POINT TO LAST CYLINDER
005522  105052                                 ADD     R5,R2           ; R2 POINTS TO LAST CYLINDER
005523  104273                          1$:    MOV     (R0)+,R3        ; GET WORD
005524  100223                                 MOV     R3,(R2)+        ; SAVE
005525  117401                                 DEC     R1              ; DECREMENT COUNT
005526  055523                                 BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
005527                                         POP     <R2,R0>         ; RESTORE
005527  104262                                                                 MOV (SP)+,R2
005530  104267                                                                 MOV (SP)+,R0
005531  104201  002201                  4$:    MOV     #CYL,R1         ; R2 POINTS TO NEW CYL
005533  104213                          2$:    MOV     (R1)+,R3        ; GET WORD
005534  100273                                 MOV     R3,(R0)+        ; SAVE
005535  117402                                 DEC     R2              ; DECREMENT COUNT
005536  055533                                 BNE     2$              ; IF COUNT INCOMPLETE, BRANCH
005537  104653  000046                         MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
005541  101203  002000                         BIS     #SEKINP,R3      ; MARK SEEK IN PROGRESS
005543  100653  000046                         MOV     R3,U.PARM(R5)   ; SAVE
005545  005551                                 BR      ISUEXT          ; BRANCH (NOTE THAT R2 IS ZERO - NO ERRORS)
005546                                  10$:   CERROR  5,#SER9         ; REPORT SECONDARY ERROR
005546  104200  005050  001110                                                 MOV     #SER9,HRQ.05
005551  000000                          ISUEXT: RETURN                 ; RETURN TO CALLING PROGRAM
                                        .IF     LE,MAXADR-.
                                  MAXADR =      .+1
                                        .ENDC
```

```
********** OVERLAY MODULE SEKTST - SEE IF THE SEEK IS COMPLETE

                                .SBTTL  ********** OVERLAY MODULE SEKTST - SEE IF THE SEEK IS COMPLETE
                                        DMOVLY  TS,BUFARA
005552  000105                          .WREDC          ;OUTPUT FDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000024                          SEKTST  =       SEEK+1          ; SEEK COMPLETE TEST OVERLAY
                                ;
                                ;       SEKTST TESTS TO SEE IF READ/WRITE READY OR ATTENTION IS HIGH
                                ;       R2 RETURNED AS ZERO IF SEEK WAS SUCESSFUL, -1 IF SEEK INCOMPLETE
                                ;       POSITIVE NON-ZERO IF AN ERROR WAS DETECTED
                                ;
                                        .ENABL  LSB
005230  020746                        1$: CALL    RTDS            ; GET REAL TIME  DRIVE STATE
005231  115002                        1$: TST     R2              ; SEE IF ERROR OCCURRED
005232  015237                        1$: BEQ     1$              ; IF NOT, BRANCH
005233  104207  000024                  MOV     #SEKTST,R0      ; RETRY THIS MODULE
005235  104051                        1$: MOV     R5,R1           ; DEFERRED CALL
005236  005617                        1$: BR      STSEXT          ; EXIT
005237  024163                  1$:   1$: CALL    ENABLE          ; ENABLE ERROR RECOVERY
005240  115001                        1$: TST     R1              ; SEE IF R/W RDY ASSERTED
005241  035417                        1$: BPL     STSERR          ; IF NOT READY, BRANCH
005242                                  ASSUME  RWRDY,100000    ; ASSUME R/W RDY IS SIGN BIT
005242  104657  000046                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005244  103207  002000                  BIC     #SEKINP,R0      ; SEEK NO LONGER IN PROGRESS
005246  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE PARAMETERS
005250  104657  000047                  MOV     U.RCOV(R5),R0   ; GET RECOVERY WORD
005252  103207  004000                  BIC     #SEKREQ,R0      ; SEEK NO LONGER REQUIRED
005254  100657  000047                  MOV     R0,U.RCOV(R5)   ; SAVE
005256  104647  000001                  MOV     S.SEEK(R4),R0   ; GET NUMBER OF SEEKS ISSUED
005260  117407                          DEC     R0              ; DECREMENT SEEK COUNT
005261  055275                          BNE     SEKCNT          ; IF NO REPORT NEEDED, BRANCH
005262  104650  000063  001104          MOV     U.UNUM(R5),HRQ.01 ; GET STARTING SUBUNIT NUMBER
005265  105650  000050  001104          ADD     U.SUBU(R5),HRQ.01 ; ADD OFFSET
005270  104207  060010                  MOV     #T4SEEK,R0      ; MOVE SEEK REPORT NUMBER TO R0
005272  021053                          CALL    HOSTRQ          ; REPORT TO HOST
005273  104207  001750                  MOV     #1000.,R0       ; SET UP FOR NEW COUNT
                                        .DSABL  LSB
005275  100647  000001          SEKCNT: MOV     R0,S.SEEK(R4)   ; SAVE COUNT
005277  114002                          CLR     R2              ; NO ERRORS
005300  104653  000007                  MOV     U.SRTY(R5),R3   ; GET RETRY COUNT
005302  015404                          BEQ     2$              ; IF NO RETRIES, BRANCH
005303                                  REPSFT  SOFT            ; REPORT SOFT ERROR
005303  104200  000001  001105                                          MOV     #1,HRQ.02
005306  114000  001106                                                  CLR     HRQ.03
005310  114000  001107                                                  CLR     HRQ.04
005312  100467                                                                  MOV R0,-(SP)
005313  104657  000063                                                  MOV     U.UNUM(R5),R0
005315  105657  000050                                                  ADD     U.SUBU(R5),R0
005317  104070  001104                                                  MOV     R0,HRQ.01
005321  104207  060007                                                  MOV     #T4SOFT,R0
005323  021053                                                          CALL    HOSTRQ
005324  104267                                                                  MOV (SP)+,R0
```

```
********** OVERLAY MODULE SEKTST - SEE IF THE SEEK IS COMPLETE

 71 005325                                  SOFTER  15,<R3,S.LETR(R4),U.CBN(R5),U.CBN+1(R5)>
    005325  104200  001503  001107                                        MOV     #ER15,HRQ.04
    005330  104030  001110                                              MOV     R3,HRQ.05
    005332  104640  000005  001111                                      MOV     S.LETR(R4),HRQ.06
    005335  104650  000053  001112                                      MOV     U.CBN(R5),HRQ.07
    005340  104650  000054  001113                                      MOV     U.CBN+1(R5),HRQ.08
    005343  104202  147657                                              MOV     #15!ERSOFT+4000.,R2
    005345  104020  001105                                              MOV     R2,HRQ.02
    005347  104200  005347  001104                                      MOV     #.,HRQ.01
    005352  104200  060013  001103                                      MOV     #ERRMES,HRQ.RQ
 72 005355                                  ERRORC  <U.PARM(R5),#RBNTXT,U.RBN(R5),U.RBN+1(R5),U.CGRP(R5),U.CCYL(R5)>
    005355  104650  000046  001114                                      MOV     U.PARM(R5),HRQ.09
    005360  104200  005472  001115                                      MOV     #RBNTXT,HRQ.10
    005363  104650  000055  001116                                      MOV     U.RBN(R5),HRQ.11
    005366  104650  000056  001117                                      MOV     U.RBN+1(R5),HRQ.12
    005371  104650  000066  001120                                      MOV     U.CGRP(R5),HRQ.13
    005374  104650  000064  001121                                      MOV     U.CCYL(R5),HRQ.14
 73 005377                                  ERRORC  U.CCYL+1(R5)
    005377  104650  000065  001122                                      MOV     U.CCYL+1(R5),HRQ.15
 74 005402                                  ENDERR  0
    005402  114000  002230                                              CLR     ERRPOS          ; CLEAR THE POSITION
 75 005404  104207  000007          2$:     MOV     #BUILDP,R0      ; BUILD THE READ/WRITE CHAIN NEXT
 76 005406  104653  000047                  MOV     U.RCOV(R5),R3   ; GET RECOVERY WORDS
 77 005410  102203  030000                  BIT     #LEVUSD!NXTLEV,R3       ; SEE IF ERROR RECOVERY LEVELS IN USE
 78 005412  015415                          BEQ     1$              ; IF NOT, BRANCH
 79 005413  104207  000017                  MOV     #NEWLEV,R0      ; ISSUE ERROR RECOVERY COMMAND NEXT
 80 005415  114001                  1$:     CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
 81 005416  005617                          BR      STSEXT          ; BRANCH
 82 005417  102201  000100          STSERR: BIT     #AVAIL,R1       ; SEE IF DRIVE TIMEOUT HAS EXPIRED
 83 005421  015433                          BEQ     1$              ; IF NOT, BRANCH
 84 005422  104653  000046                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
 85 005424  103203  002000                  BIC     #SEKINP,R3      ; MARK AS SEEK NOT IN PROGRESS
 86 005426  100653  000046                  MOV     R3,U.PARM(R5)   ; SAVE
 87 005430  104207  000023                  MOV     #SEEK,R0        ; SEEK AGAIN
 88 005432  005617                          BR      STSEXT          ; EXIT
 89 005433  102201  000002          1$:     BIT     #ATTN,R1        ; SEE IF ATTENTION ASSERTED
 90 005435  015527                          BEQ     2$              ; IF NOT, BRANCH
 91 005436                                  REPSFT  SOFT,,SEEK      ; REPORT SEEK AND SOFT ERROR
    005436  104200  000001  001105                                      MOV     #1,HRQ.02
    005441  114000  001106                                              CLR     HRQ.03
    005443  104200  000001  001107                                      MOV     #1,HRQ.04
    005446  100467                                                              MOV R0,-(SP)
    005447  104657  000063                                              MOV     U.UNUM(R5),R0
    005451  105657  000050                                              ADD     U.SUBU(R5),R0
    005453  104070  001104                                              MOV     R0,HRQ.01
    005455  104207  060007                                              MOV     #T4SOFT,R0
    005457  021053                                                      CALL    HOSTRQ
    005460  104267                                                              MOV (SP)+,R0
 92 005461                                  SOFTER  1,<U.LGRP(R5),U.LCYL(R5),U.LCYL+1(R5),U.CGRP(R5),U.CCYL(R5)>
    005461  104200  000000  001107                                      MOV     #ER1,HRQ.04
    005464  104650  000071  001110                                      MOV     U.LGRP(R5),HRQ.05
    005467  104650  000067  001111                                      MOV     U.LCYL(R5),HRQ.06
    005472  104650  000070  001112                                      MOV     U.LCYL+1(R5),HRQ.07
    005475  104650  000066  001113                                      MOV     U.CGRP(R5),HRQ.08
    005500  104650  000064  001114                                      MOV     U.CCYL(R5),HRQ.09
    005503  104202  147641                                              MOV     #1!ERSOFT+4000.,R2
    005505  104020  001105                                              MOV     R2,HRQ.02
```

```
********** OVERLAY MODULE SEKTST - SEE IF THE SEEK IS COMPLETE

        005507  104200  005507  001104                                          MOV     #.,HRQ.01
        005512  104200  060013  001103                                          MOV     #ERRMES,HRQ.RQ
     93 005515                          ERRORC  U.CCYL+1(R5)
        005515  104650  000065  001115                                          MOV     U.CCYL+1(R5),HRQ.10
     94 005520                          ENDERR                  ; REPORT ATTN ASSERTED DURING SEEK
        005520  104200  000051  001116                                          MOV     #SER22,HRQ.11
        005523  104200  000014  002230                                          MOV     #11+1,ERRPOS    ; SET THE POSITION
     95 005526  005617                  BR      STSEXT          ; EXIT
     96 005527  024172              2$: CALL    DSABLE
     97 005530  104051                  MOV     R5,R1           ; MAKE R1 NON-ZERO FOR DEFERRED CALL
     98                             ;   NOTE:   R2 IS ZERO AT THIS POINT
     99 005531  104207  000024          MOV     #SEKTST,R0      ; SEEK TEST IS NEXT MODULE
    100 005533  104653  000006          MOV     U.TIML(R5),R3   ; GET TIMEOUT VALUE
    101 005535  107203  000001          SUB     #1,R3           ; DECREMENT TIMEOUT VALUE
    102 005537  100653  000006          MOV     R3,U.TIML(R5)   ; SAVE TIMEOUT VALUE
    103 005541  045617                  BCC     STSEXT          ; IF NON-ZERO BRANCH
    104 005542  104653  000005          MOV     U.TIMH(R5),R3   ; GET HI ORDER TIMEOUT
    105 005544  107203  000001          SUB     #1,R3           ; DECREMENT COUNT
    106 005546  100653  000005          MOV     R3,U.TIMH(R5)   ; SAVE
    107 005550  045617                  BCC     STSEXT
    108 005551                          SOFTER  3,<U.LGRP(R5),U.LCYL(R5),U.LCYL+1(R5)>
        005551  104200  000166  001107                                          MOV     #ER3,HRQ.04
        005554  104650  000071  001110                                          MOV     U.LGRP(R5),HRQ.05
        005557  104650  000067  001111                                          MOV     U.LCYL(R5),HRQ.06
        005562  104650  000070  001112                                          MOV     U.LCYL+1(R5),HRQ.07
        005565  104202  147643                                                  MOV     #3!ERSOFT+4000.,R2
        005567  104020  001105                                                  MOV     R2,HRQ.02
        005571  104200  005571  001104                                          MOV     #.,HRQ.01
        005574  104200  060013  001103                                          MOV     #ERRMES,HRQ.RQ
    109 005577                          ERRORC  <U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
        005577  104650  000066  001113                                          MOV     U.CGRP(R5),HRQ.08
        005602  104650  000064  001114                                          MOV     U.CCYL(R5),HRQ.09
        005605  104650  000065  001115                                          MOV     U.CCYL+1(R5),HRQ.10
    110 005610                          ENDERR
        005610  104200  000051  001116                                          MOV     #SER22,HRQ.11
        005613  104200  000014  002230                                          MOV     #11+1,ERRPOS    ; SET THE POSITION
    111 005616  024207                  CALL    GOSEEK
    112 005617  003231          STSEXT: BR      JMPRET          ; RETURN TO SEQUENCER
    116                                 .IF     LE,MAXADR-.
    117                         MAXADR  =       .+1
    118                                 .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE RECALB - RECALIBRATION
005620                                  DMOVLY  RC,BUFARA
005620  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000025                          RECALB  =       SEKTST+1                ; RECALIBRATION MODULE
                                ;
                                ;
                                ;
                                        .ENABL  LSB
005230  104203  001655                  MOV     #CR.INR,R3      ; POINT TO THE INITIATE RECALIBRATION COMMAND
005232  021173                          CALL    TALK            ; SEND-RECEIVE SDI COMMAND
005233  115002                          TST     R2              ; SEE IF ANY ERRORS OCCURRED
005234  015264                          BEQ     1$              ; IF NOT, BRANCH
                                ;
                                ;       FROM HERE TO THE LABEL '5$' IS CODE THAT SEES IF THE HOST WAS
                                ;       HALTED LONG ENOUGH FOR THE DRIVE TO GO OFFLINE DURING THE RECEIVE
                                ;       LEVEL 2 OPERATION (NOTE THAT LONG TIMEOUTS TALK TO THE HOST <<DURING>>
                                ;       THE TIMEOUT
                                ;
005235  106202  147747                  CMP     #71.!ERSOFT+4000.,R2    ; SEE IF TIMEOUT OF RECEIVE
005237  055260                          BNE     5$              ; IF NOT, BRANCH
005240  020746                          CALL    RTDS            ; GET STATE
005241  115002                          TST     R2              ; SEE IF ERROR
005242  015245                          BEQ     4$              ; IF NOT, BRANCH
005243  024201                          CALL    GORTRY          ; RETRY THIS MODULE
005244  005306                          BR      2$              ; EXIT
005245  102201  000100          4$:     BIT     #AVAIL,R1       ; SEE IF DRIVE IS AVAILABLE
005247  015257                          BEQ     6$              ; IF NOT, BRANCH
005250  104303  001661                  MOV     CR.INR+L2.EOF,R3        ; GET INR OFFSET
005252  105053                          ADD     R5,R3           ; POINT TO RECALIBRATE ERROR COUNT
005253  104131                          MOV     (R3),R1         ; GET COUNT
005254  117401                          DEC     R1              ; DECREMENT COUNT
005255  100131                          MOV     R1,(R3)         ; SAVE
005256  005274                          BR      3$              ;
005257  104052                  6$:     MOV     R5,R2           ; FLAG ERROR
005260                          5$:     CERROR  5,#SER7         ; REPORT SECONDARY ERROR
005260  104200  005010  001110                                                  MOV     #SER7,HRQ.05
005263  005306                          BR      2$              ; EXIT
005264  104653  000047          1$:     MOV     U.RCOV(R5),R3   ; GET RECOVERY PARAMETERS
005266  103203  002000                  BIC     #RCBREQ,R3      ; CLEAR RECALIBRATION REQUESTED FLAG
005270  101203  004000                  BIS     #SEKREQ,R3      ; MARK SEEK AS REQUIRED
005272  100653  000047                  MOV     R3,U.RCOV(R5)   ; SAVE
005274  114002                  3$:     CLR     R2              ; NO ERRORS
005275  100652  000064                  MOV     R2,U.CCYL(R5)   ; ZERO CYLINDER
005277  100652  000065                  MOV     R2,U.CCYL+1(R5)
005301  100652  000066                  MOV     R2,U.CGRP(R5)
005303  114001                          CLR     R1              ; IMMIDATE CALL
005304  104207  000023                  MOV     #SEEK,R0        ; SEEK IS NEXT MODULE
005306  003231                  2$:     BR      JMPRET          ; RETURN TO SEQNCR
                                        .DSABL  LSB
                                        .IF     LE,MAXADR-.
```

```
                                MAXADR  =       .+1
                                        .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE DRPALL - DROP ALL SUBUNITS ON THIS DRIVE
005307                                  DMOVLY  DA,AREA0
005307  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
000026                          DRPALL  =       RECALB+1
                                ;
                                ;       DROP ALL SUBUNITS ON THIS DRIVE AND REPORT
                                ;
                                ;       NOTE:
                                ;               THIS IS A VERY UNSTRUCTURED PIECE OF CODE -- IT HAS SEVERAL
                                ;               JUMPS DIRECTLY INTO SEQNCR AT SEVERAL DIFFERENT PLACES.
                                ;
                                        .ENABL  LSB
004413                                  PUSH    R4              ; SAVE R4 (JUST IN CASE)
004413  100464                                                                  MOV R4,-(SP)
004414  104200  005545  001105        MOV     #MS2,HRQ.02     ; POINT TO MESSAGE
004417  104207  001107                MOV     #HRQ.04,R0      ; POINT TO WHERE TO PUT UNITS
004421  104053                        MOV     R5,R3           ; GET POINTER TO UNIT DATABASE
004422  115403                        INC     R3              ; POINT TO SUBUNIT POINTERS
004423                                ASSUME  U.SUBP,1
004423  114001                        CLR     R1              ; CLEAR INDEX
004424  114004                        CLR     R4              ; CLEAR NUMBER OF SUBUNITS COUNT
004425  104232                1$:     MOV     (R3)+,R2        ; GET POINTER
004426  074436                        BMI     2$              ; IF NO SUBUNIT, BRANCH
004427  104652  000063                MOV     U.UNUM(R5),R2   ; GET BASE UNIT NUMBER
004431  105012                        ADD     R1,R2           ; ADD OFFSET
004432  100272                        MOV     R2,(R0)+        ; MOVE TO OUTPUT BUFFER
004433  104020  001104                MOV     R2,HRQ.01       ; MOVE TO UNIT NUMBER
004435  115404                        INC     R4              ; INC SUBUNIT COUNT
004436  115401                2$:     INC     R1              ; INC COUNT
004437  106201  000003                CMP     #3,R1           ; SEE IF EXPIRED
004441  044425                        BCC     1$              ; IF NOT, BRANCH
004442  104647  004460                MOV     SER18F-1(R4),R0 ; POINT TO DRIVE LIST
004444  104070  001106                MOV     R0,HRQ.03       ; POINT TO CORRECT ERROR MESSAGE
004446  104207  060015                MOV     #MESSAG,R0      ; MESSAGE TO R0
004450  021053                        CALL    HOSTRQ          ; REPORT
004451  104651  000046                MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
004453  101201  100000                BIS     #DROP,R1        ; SET DROP BIT
004455  100651  000046                MOV     R1,U.PARM(R5)   ; SAVE
004457                                POP     R4
004457  104264                                                                  MOV (SP)+,R4
004460  003675                        BR      NOSUB
                                ;
004461  005432                SER18F: .WORD   SER18A
004462  005426                        .WORD   SER18B
004463  005422                        .WORD   SER18C
004464  005416                        .WORD   SER18D
                                        .DSABL  LSB
                                ;
                                        .IF     LE,MAXADR-.
                                MAXADR  =       .+1
```

```
     75                                          .ENDC
```

```
      1                                 .SBTTL  ********** OVERLAY MODULE INSET - SET UP UNIT FOR INITIALZATION
      5 004465                                  DMOVLY  IN,AREAI
        004465  000105                         .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
      9                                 ;*****************************************************************************
     10                                 ;*****************************************************************************
     11                                 ;*****************************************************************************
     12                                 ;*****************************************************************************
     13                                 ;
     14                                 ;
     15                                 ;
     19           000027                INSET   =       DRPALL+1
     28                                 ;
     29                                 ;       INSET WILL SET UP EACH UNIT BEFORE IT STARTS RUNNING
     30                                 ;
     31                                         .ENABL  LSB
     32 004701   104657  000046                 MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
     33 004703   103207  004000                 BIC     #NEWSUB,R0      ; NO LONGER NEW SUBUNITS
     34 004705   100657  000046                 MOV     R0,U.PARM(R5)   ; SAVE
     35 004707   114002                         CLR     R2              ; NO ERRORS
     36 004710   100652  000064                 MOV     R2,U.CCYL(R5)   ; CLEAR CYL+GROUP
     37 004712   100652  000065                 MOV     R2,U.CCYL+1(R5)
     38 004714   100652  000066                 MOV     R2,U.CGRP(R5)
     39 004716   114001                         CLR     R1              ; IMMIDATE CALL
     40 004717   104207  000030                 MOV     #COMCHR,R0      ; COMMON CHARACTERISTICS NEXT MODULE CALLED
     41 004721   003231                         BR      JMPRET          ; RETURN TO SEQUENCER
     42                                         .DSABL  LSB
     46                                         .IF     LE,MAXADR-.
     47                                 MAXADR  =       .+1
     48                                         .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE COMCHR - SET UP COMMON CHARACTERISTICS
                                        DMOVLY  CC,AREAI
004722  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000030                          COMCHR  =       INSET+1
                                ;
                                ;       GET COMMON CHARACTERISTICS AND SET UP THE ERROR RECOVERY LEVEL,
                                ;       RETRIES, LONG TIMEOUT, AND SHORT TIMEOUT
                                ;
                                        .ENABL  LSB
004701  104203  004453                  MOV     #CR.GCR,R3      ; POINT TO GET CHARACTERISTICS DATA BLOCK
004703  021173                          CALL    TALK            ; GET CHARACTERISTICS
004704  115002                          TST     R2              ; SEE IF ANY ERRORS OCCURRED
004705  014723                          BEQ     1$              ; IF NOT, BRANCH
004706                                  CERROR  5,#SER8         ; REPORT SECONDARY ERROR
004706  104200  005027  001110                                          MOV     #SER8,HRQ.05
004711  103200  100000  001105          BIC     #C2DFTL,HRQ.02  ; CHANGE TO HARD ERROR
004714  104200  060014  001103          MOV     #ERRMC,HRQ.RQ   ; COUNT ERROR
004717  101200  000002  002223          BIS     #DIE,M.PARM     ; FLAG INITIALIZATION ERROR
004722  005027                          BR      2$              ; BRANCH TO EXIT
004723  104300  001706  004516  1$:     MOV     ST+DRVID,DSERNM ; SAVE DRIVE SERIAL NUMBER
004726  104300  001707  004517          MOV     ST+DRVID+1,DSERNM+1     ; SAVE DRIVE SERIAL NUMBER
004731  104300  001710  004520          MOV     ST+DRVID+2,DSERNM+2     ; SAVE DRIVE SERIAL NUMBER
004734  104307  001703                  MOV     ST+RETS,R0      ; GET NUMBER OF RETRIES ALLOWED
004736  110607                          ROR     R0              ; ROTATE TO CORRECT POSITION
004737  110607                          ROR     R0
004740  110607                          ROR     R0
004741  110607                          ROR     R0
004742  103207  177760                  BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
004744  100657  000030                  MOV     R0,U.RTRY(R5)   ; SAVE IN UNIT PARAMETERS
004746  104307  001704                  MOV     ST+ERLEV,R0     ; GET ERROR RECOVERY LEVELS
004750  103207  177400                  BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
004752  100657  000031                  MOV     R0,U.MLEV(R5)   ; SAVE IN UNIT PARAMETERS
004754  104307  001704                  MOV     ST+ECCRSH,R0    ; GET ECC THRESHOLD
004756  110707                          SWAB    R0              ; MOVE ECC THRESHOLD TO LOWER BYTE
004757  103207  177400                  BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
004761  100657  000032                  MOV     R0,U.ECCT(R5)   ; SAVE
004763  104307  001703                  MOV     ST+LONGTO,R0    ; GET LONG TIMEOUT
004765  103207  177760                  BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
004767  025031                          CALL    TO              ; CALCULATE TIMEOUT
004770  100657  000034                  MOV     R0,U.SDIL(R5)   ; SAVE IN UNIT PARAMETERS
004772  104307  001702                  MOV     ST+SHRTTO,R0    ; GET SHORT TIMEOUT
004774  103207  177760                  BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
004776  025031                          CALL    TO              ; CALCULATE TIMEOUT
004777  100657  000033                  MOV     R0,U.SDIS(R5)   ; SAVE SHORT TIMEOUT IN UNIT PARAMETERS
005001  114001                          CLR     R1              ; CLEAR LO SEEK TIMEOUT
005002  114002                          CLR     R2              ; CLEAR HI TIMEOUT
005003  105201  025762          4$:     ADD     #11250.,R1      ; ADD 9 SEC TIMEOUT TO LO ORDER TIMEOUT
005005  045007                          BCC     3$              ; IF NO CARRY, BRANCH
005006  115402                          INC     R2              ; PROPOGATE CARRY
005007  117407          3$:             DEC     R0              ; DECREMENT TIMEOUT
```

```
     73 005010  055003                          BNE     4$              ; IF UNEXPIRED, BRANCH
     74 005011  115402                          INC     R2              ; ROUND UP
     75 005012  100652  000011                  MOV     R2,U.MSTO(R5)   ; SAVE SEEK MASTER TIMEOUT
     76 005014  104307  001703                  MOV     ST+RCTCPS,R0    ; GET NUMBER OF RCT COPIES
     77 005016  110707                          SWAB    R0              ; MOVE TO LOW WORD
     78 005017  103207  177760                  BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
     79 005021  117407                          DEC     R0              ; ADJUST FOR TEST 4 INTERNALS
     80 005022  100657  000057                  MOV     R0,U.COPY(R5)   ; SAVE
     81 005024  104207  000031                  MOV     #SPINUP,R0      ; SPINUP NEXT MODULE
     82 005026  114002                          CLR     R2              ; NO ERRORS
     83 005027  114001                  2$:     CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
     84 005030  003231                          BR      JMPRET          ; RETURN TO SEQUNCER
     85                                         .DSABL  LSB
     86 005031                          TO:
     87                                 ;
     88                                 ;       CALCULATE THE TIMEOUT IN 9SEC INTERVALS (UDA SDI RECEIVE XFC TAKES
     89                                 ;       9 SEC)
     90                                 ;
     91 005031  104201  000001                  MOV     #1,R1           ; SET UP LOG2 SHIFTER
     92 005033  105011                  1$:     ADD     R1,R1           ; DOUBLE THE TIMEOUT VALUE
     93 005034  117407                          DEC     R0              ; DECREMENT COUNT
     94 005035  055033                          BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
     95 005036  115407                  2$:     INC     R0              ; INCREMENT 9 SEC COUNT
     96 005037  107201  000011                  SUB     #9.,R1          ; SUBTRACT 9 SEC FROM TIMEOUT
     97 005041  035036                          BPL     2$              ; IF MORE TIME TO GO, BRANCH
     98 005042  000000                          RETURN                  ; RETURN TO CALLING PROGRAM
    102                                         .IF     LE,MAXADR-.
    103                                 MAXADR  =       .+1
    104                                         .ENDC
```

```
********** OVERLAY MODULE SPINUP - SPIN THE DRIVE UP

                                        .SBTTL  ********** OVERLAY MODULE SPINUP - SPIN THE DRIVE UP
                                                DMOVLY  SP,AREAI
005043  000105                                  .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;
                                        ;
                                        ;
000031                                  SPINUP  =       COMCHR+1
                                        ;
                                        ;       SPINUP WILL SPIN THE DRIV UP IF SPUN DOWN
                                        ;
                                                .ENABL  LSB
004701  104203  004465                          MOV     #CR.RUN,R3      ; POINT TO RUN COMMAND
004703  021173                                  CALL    TALK            ; SEND COMMAND
004704  115002                                  TST     R2              ; SEE IF ERROR OCCURRED
004705  054711                                  BNE     1$              ; IF SO, BRANCH
004706  104207  000032                          MOV     #SORT,R0        ; SORT NEXT MODULE
004710  004725                                  BR      2$              ; EXIT
004711                                  1$:     CERROR  5,#SER6         ; REPORT SECONDARY ERROR
004711  104200  004770  001110                                                  MOV     #SER6,HRQ.05
004714  103200  100000  001105                  BIC     #C2DFTL,HRQ.02  ; CHANGE TO HARD ERROR
004717  104200  060014  001103                  MOV     #ERRMC,HRQ.RQ   ; COUNT ERROR
004722  101200  000002  002223                  BIS     #DIE,M.PARM     ; FLAG INITIALIZATION ERROR
004725  114001                          2$:     CLR     R1              ; IMMIDATE CALL
004726  003231                                  BR      JMPRET          ; RETURN TO SEQNCR
                                                .DSABL  LSB
                                                .IF     LE,MAXADR-.
                                        MAXADR  =       .+1
                                                .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE SORT - SORT ALL CYLINDERS, BEGIN/SETS, AND BAD BLOCKS
 004727                                 DMOVLY  SO,AREAI
 004727  000105                         .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
         000032                 SORT    =       SPINUP+1
                                ;
                                ;       SORT WILL SORT ALL BEGIN/END SETS, BAD BLOCKS AND TRACKS/GROUPS IN
                                ;       ASCENDING ORDER
                                ;
                                        .ENABL  LSB
 004701                                 PUSH    <R4,R5>         ; SAVE R4 AND R5
 004701  100464                                                                 MOV R4,-(SP)
 004702  100465                                                                 MOV R5,-(SP)
 004703  115405                         INC     R5              ; POINT TO SUBUNIT POINTERS
 004704                                 ASSUME  U.SUBP,1
 004704  104204  000004                 MOV     #4,R4           ; MAXIMUM OF 4 SUBUNITS
 004706  104257                 4$:     MOV     (R5)+,R0        ; R0 POINTS TO SUBUNIT DATABASE
 004707  074724                         BMI     3$              ; IF NO SUBUNIT, BRANCH
 004710                                 PUSH    <R4,R5>         ; SAVE R4 AND R5
 004710  100464                                                                 MOV R4,-(SP)
 004711  100465                                                                 MOV R5,-(SP)
 004712  104171                         MOV     (R0),R1         ; GET SUBUNIT PARAMETERS
 004713                                 ASSUME  S.PARM,0
 004713  102201  000040                 BIT     #BEUSED,R1      ; SEE IF BEGIN/END SETS USED
 004715  054720                         BNE     1$              ; IF SO, BRANCH
 004716  025074                         CALL    SORTTG          ; SORT THE TRACKS/GROUPS
 004717  004721                         BR      2$              ; BRANCH
 004720  024735                 1$:     CALL    SORTBE          ; SORT THE BEGIN/END SETS
 004721  025025                 2$:     CALL    SORTBB          ; SORT THE BAD BLOCKS
 004722                                 POP     <R5,R4>         ; RESTORE R5, R4
 004722  104265                                                                 MOV (SP)+,R5
 004723  104264                                                                 MOV (SP)+,R4
 004724  117404                 3$:     DEC     R4              ; DECREMENT COUNT
 004725  054706                         BNE     4$              ; IF INCOMPLETE, BRANCH
 004726                                 POP     <R5,R4>         ; RESTORE R5, R4
 004726  104265                                                                 MOV (SP)+,R5
 004727  104264                                                                 MOV (SP)+,R4
 004730  114001                         CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
 004731  114002                         CLR     R2              ; NO ERRORS
 004732  104207  000033                 MOV     #SCHAR0,R0      ; SCHAR0 NEXT MODULE
 004734  003231                         BR      JMPRET
                                        .DSABL  LSB
```

```
                                        .SBTTL  SORTBE - SORT THE BEGIN/END SETS IN ASCENDING ORDER
004735                                  SORTBE:
                                        ;
                                        ;       SORT THE BEGIN/END SETS IN ASCENDING ORDER
                                        ;
004735  104672  000016                          MOV     S.BESS+3(R0),R2 ; SEE IF ONLY ONE BEGIN END SET
004737  074774                                  BMI     4$              ; IF SO, EXIT (ALLREADY SORTED)
004740  104202  000013                          MOV     #S.BESS,R2      ; R2 WILL POINT TO START OF BEGIN/END SETS
004742  105072                                  ADD     R0,R2           ; R2 POINTS TO BEGIN/END SETS
004743  104023                          1$:     MOV     R2,R3           ; R3 WILL POINT TO NEXT BEGIN/END SET
004744  105203  000004                 2$:     ADD     #4,R3           ; R3 POINTS TO BEGIN/END SETS
                                        ;
                                        ;       28 BIT COMPARE FOR BEGIN/END SET SORTING
                                        ;
004746  104634  000003                          MOV     3(R3),R4        ; GET WORD THAT R3 POINTS TO
004750  103204  170000                          BIC     #^CHBHINB,R4    ; STRIP OFF UNUSED BITS
004752  104625  000003                          MOV     3(R2),R5        ; GET OTHER WORD TO COMPARE
004754  106045                                  CMP     R4,R5           ; COMPARE
004755  054762                                  BNE     10$             ; IF DIFFERENCE IS FOUND, BRANCH
004756  104625  000002                          MOV     2(R2),R5        ; GET OTHER WORD TO COMPARE
004760  106635  000002                          CMP     2(R3),R5        ; COMPARE
004762  044764                          10$:    BCC     3$              ; IF R2->B/E <= R3->B/E THEN ALLREADY IN ORDER
004763  024775                                  CALL    SWAPBE          ; SWAP TE BEGIN/END SETS
004764  104635  000003                 3$:     MOV     3(R3),R5        ; SEE IF R3->END-OF-LIST
004766  034744                                  BPL     2$              ; IF NOT, BRANCH
004767  105202  000004                          ADD     #4,R2           ; R2->NEXT BEGIN/END SET
004771  104625  000003                          MOV     3(R2),R5        ; SEE IF R2->END-OF-LIST
004773  034743                                  BPL     1$              ; IF NOT, BRANCH
004774  000000                          4$:     RETURN                  ; RETURN TO COPYSU
```

```
SWAPBE - IF BEGIN/END SETS OUT OF ORDER, SWAP

                                .SBTTL  SWAPBE - IF BEGIN/END SETS OUT OF ORDER, SWAP
004775                          SWAPBE:
                                ;
                                ;       SWAP THE BEGIN/END SET POINTED TO BY R2 WITH THE BEGIN/END SET
                                ;       POINTED TO BY R3, KEEPING THE END-OF-LIST FLAG INTACT
                                ;
004775                                  PUSH    <R1,R2,R3>      ; SAVE POINTERS
004775  100461                                                          MOV R1,-(SP)
004776  100462                                                          MOV R2,-(SP)
004777  100463                                                          MOV R3,-(SP)
005000  104201  000003                   MOV     #3,R1           ; SET UP LOOP COUNT
005002  104124                  1$:     MOV     (R2),R4         ; GET WORD FROM SET
005003  104135                          MOV     (R3),R5         ; GET WORD FROM OTHER SET
005004  100234                          MOV     R4,(R3)+        ; SWAP WORD
005005  100225                          MOV     R5,(R2)+        ; SWAP WORD
005006  117401                          DEC     R1              ; DECREMENT COUNT
005007  055002                          BNE     1$              ; LOOP IF INCOMPLETE
005010  104124                          MOV     (R2),R4         ; GET WORD FROM SET
005011  104135                          MOV     (R3),R5         ; GET WORD FROM OTHER SET
005012  104051                          MOV     R5,R1           ; MOVE TO TEMP STORAGE
005013  103205  177400                  BIC     #HIBYTE,R5      ; STRIP OFF END-OF-LIST FLAG, IF ANY
005015  107051                          SUB     R5,R1           ; R1 CONTAINS END-OF-LIST FLAG, IF ANY
005016  101014                          BIS     R1,R4           ; R4 NOW HAS END-OF-LIST FLAG, IF ANY
005017  100134                          MOV     R4,(R3)         ; SWAP WORD (AND EOL FLAG, IF ANY)
005020  100125                          MOV     R5,(R2)         ; SWAP WORD
005021                                  POP     <R3,R2,R1>      ; RESTORE THE REGISTERS
005021  104263                                                          MOV (SP)+,R3
005022  104262                                                          MOV (SP)+,R2
005023  104261                                                          MOV (SP)+,R1
005024  000000                          RETURN                  ; RETURN TO SORTBE
```

SORTBB - SORT THE BAD BLOCKS IN ASCENDING ORDER

```
                                        .SBTTL  SORTBB - SORT THE BAD BLOCKS IN ASCENDING ORDER
005025                                  SORTBB:
                                        ;
                                        ;       SORT THE BAD BLOCKS IN ASCENDING ORDER
                                        ;
005025  104672  000012                  MOV     S.BADP(R0),R2   ; GET THE BAD BLOCK POINTER
005027  015073                          BEQ     4$              ; IF NO BAD BLOCKS, BRANCH
005030  104625  000001                  MOV     1(R2),R5        ; SEE IF ONLY ONE BAD BLOCK
005032  075073                          BMI     4$              ; IF SO BRANCH (ALLREADY SORTED)
005033  104023                  1$:     MOV     R2,R3           ; R3 WILL POINT AT NEXT BAD BLOCK
005034  105203  000002          2$:     ADD     #2,R3           ; R3 POINTS TO NEXT BAD BLOCK
005036  024623                          CALL    CBB2            ; COMPARE THE TWO BLOCKS
005037  045063                          BCC     3$              ; IF IN ASCENDING ORDER, BRANCH
                                        .SBTTL  SWAPBB - IF BAD BLOCKS OUT OF ORDER, SWAP
                                        ;SWAPBB
                                        ;
                                        ;       SWAP THE BAD BLOCKS, RETAINING THE END-OF-LIST POINTER AT THE END
                                        ;
005040                                  PUSH    R1              ; SAVE R1
005040  100461                                                          MOV R1,-(SP)
005041  104124                          MOV     (R2),R4         ; GET LO ORDER BAD BLOCK OF 1ST SET
005042  104135                          MOV     (R3),R5         ; GET LO ORDER BAD BLOCK OF 2ND SET
005043  100134                          MOV     R4,(R3)         ; SWAP
005044  100125                          MOV     R5,(R2)         ; SWAP
005045  104624  000001                  MOV     1(R2),R4        ; GET HI ORDER BAD BLOCK OF 1ST SET
005047  104635  000001                  MOV     1(R3),R5        ; GET HI ORDER BAD BLOCK OF 2ND SET
005051  104051                          MOV     R5,R1           ; MOVE TO R1
005052  103205  177400                  BIC     #HIBYTE,R5      ; STRIP OFF END-OF-LIST FLAG, IF ANY
005054  107051                          SUB     R5,R1           ; R1 CONTAINS END-OF-LIST FLAG, IF ANY
005055  101014                          BIS     R1,R4           ; PUT END-OF-LIST FLAG IN R4, IF ANY
005056  100634  000001                  MOV     R4,1(R3)        ; SWAP
005060  100625  000001                  MOV     R5,1(R2)        ; SWAP
005062                                  POP     R1              ; RESTORE R1
005062  104261                                                          MOV (SP)+,R1
005063  104635  000001          3$:     MOV     1(R3),R5        ; SEE IF END-OF-LIST
005065  035034                          BPL     2$              ; IF NOT, BRANCH
005066  105202  000002                  ADD     #2,R2           ; R2 POINTS TO NEXT BAD BLOCK
005070  104625  000001                  MOV     1(R2),R5        ; SEE IF END-OF-LIST
005072  035033                          BPL     1$              ; IF NOT, BRANCH
005073  000000                  4$:     RETURN                  ; RETURN TO COPYSU
```

```
                                        .SBTTL  SORTTG - SORT THE TRACK/GROUPS IN ASCENDING ORDER
005074                                  SORTTG:
                                        ;
                                        ;       SORTTG WILL SORT THE TRACKS OR GROUPS IN ASCENDING ORDER
                                        ;
005074  104201  000017                        MOV     #S.TGSS,R1      ; R1 WILL POINT TO TRACK/GROUP START
005076  105071                                ADD     R0,R1           ; R1 POINTS TO TRACK/GROUP START
005077  104115                          1$:   MOV     (R1),R5         ; GET START OF LIST
005100  075121                                BMI     4$              ; IF NEGATIVE, WHOLE LIST IS SORTED, EXIT
005101  104012                                MOV     R1,R2           ; R2 WILL POINT TO NEXT LIST MEMBER
005102  115402                          2$:   INC     R2              ; R2 POINTS TO NEXT MEMBER
005103  104125                                MOV     (R2),R5         ; GET NEXT MEMBER
005104  104054                                MOV     R5,R4           ; COPY TO R4
005105  103205  177400                        BIC     #HIBYTE,R5      ; CLEAR END-OF-LIST FLAG (IF ANY)
005107  107054                                SUB     R5,R4           ; SAVE END-OF-LIST FLAG (IF ANY)
005110  106115                                CMP     (R1),R5         ; SEE IF THE MEMBERS ARE IN ORDER
005111  075115                                BMI     3$              ; IF SO, BRANCH
005112  101114                                BIS     (R1),R4         ; COPY START OF LIST TO R4, RETAINING EOL FLAG
005113  100115                                MOV     R5,(R1)         ; SWAP
005114  100124                                MOV     R4,(R2)         ; SWAP
005115  104125                          3$:   MOV     (R2),R5         ; GET MEMBER THAT SORT POINTER POINTS TO
005116  035102                                BPL     2$              ; IF NOT END-OF-LIST, BRANCH
005117  115401                                INC     R1              ; POINT TO NEXT START OF LIST
005120  005077                                BR      1$              ; LOOP
005121  000000                          4$:   RETURN                  ; RETURN TO CALLING PROGRAM
                                        .IF     LE,MAXADR-.
                                MAXADR  =       .+1
                                        .ENDC
```

```
********** OVERLAY MODULE SCHARO - SET UP SUBUNITS, CHECK THAT

                                .SBTTL  ********** OVERLAY MODULE SCHARO - SET UP SUBUNITS, CHECK THAT ALL PARAMETERS ARE WI
                                        DMOVLY  S0,AREAI
005122  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000033                          SCHARO  =       SORT+1
                                ;
                                ;       INSCHR WILL GET THE SUBUNIT CHARACTERISTICS AND INITILIZE THE
                                ;       SUBUNIT PARAMETERS
                                ;
                                        .ENABL  LSB
004701  100464                  PUSH    R4              ; SAVE R4 (SUBUNIT) POINTER
                                                                        MOV R4,-(SP)
004702  104041                  MOV     R4,R1           ; R1 POINTS TO SUBUNIT
004703  104657  000050          MOV     U.SUBU(R5),R0   ; GET SUBUNIT OFFSET
                                .SBTTL  SMASK  - CALCULATE THE SUBUNIT MASK
                                ;SMASK
                                ;
                                ;       SMASK TAKES THE UNIT OFFSET (0 - 3) IN R0 AND CHANGES IT TO THE
                                ;       SUBUNIT MASK (0001 - 1000)
                                ;
004705  100461                  PUSH    R1              ; SAVE R1
                                                                        MOV R1,-(SP)
004706  104071                  MOV     R0,R1           ; MOVE R0 TO R1
004707  104207  000020          MOV     #20,R0          ; SUBUNIT 0 MASK
004711  115001                  TST     R1              ; SEE IF SUBUNIT MASK SHIFTED TO CORRECT POSITION
004712  014716          SMASKL: BEQ     SMASKX          ; IF SO, BRANCH
004713  110207                  ROL     R0              ; SHIFT MASK
004714  117401                  DEC     R1              ; DECREMENT COUNT
004715  004712                  BR      SMASKL          ; BRANCH
004716  104261          SMASKX: POP     R1              ; RESTORE R1
                                                                        MOV (SP)+,R1
004717  104070  004475          MOV     R0,SUBUNT       ; MOVE TO SDI BUFFER
004721  104203  004460          MOV     #CR.SCR,R3      ; POINT TO COMMAND
004723  100461                  PUSH    R1              ; SAVE R1
                                                                        MOV R1,-(SP)
004724  021173                  CALL    TALK            ; SDI EXCHANGE
004725  104261                  POP     R1              ; RESTORE R1
                                                                        MOV (SP)+,R1
004726  115002                  TST     R2              ; SEE IF ERROR OCCURRED
004727  055135                  BNE     13$             ; IF SO, BRANCH
004730  100612  000010          MOV     R2,S.MEGR(R1)   ; ZERO MEGABIT COUNT
004732  100612  000011          MOV     R2,S.MEGW(R1)   ; ZERO MEGABIT COUNT
004734  104203  001702          MOV     #ST,R3          ; R3 POINTS TO SUBUNIT CHARACTERISTCS
004736  106205  007702          CMP     #FIRSTU,R5      ; IS R5 -> FIRST UNIT?
004740  014754                  BEQ     1$              ; IF SO, BRANCH
004741  104202  007702          MOV     #FIRSTU,R2      ; R2 -> FIRST UNIT
004743  100461                  PUSH    <R1,R3,R4>      ; SAVE REGS
                                                                        MOV R1,-(SP)
004744  100463                                                          MOV R3,-(SP)
004745  100464                                                          MOV R4,-(SP)
004746  025156                  CALL    TRAV            ; TRAVERSE THE UNITS TO FIND IF SUBUNIT CHAR ARE EQUAL
```

```
     66 004747                                          POP     <R4,R3,R1>      ; RESTORE REGS
        004747  104264                                                                  MOV (SP)+,R4
        004750  104263                                                                  MOV (SP)+,R3
        004751  104261                                                                  MOV (SP)+,R1
     67 004752  115007                                  TST     R0              ; WERE ANY CHARS EQUAL?
     68 004753  054767                                  BNE     3$              ; IF SO, BRANCH (R0 WILL BE NONE ZERO IF MATCH)
     69 004754  104207  000023              1$:         MOV     #19.,R0         ; MOVE WORD COUNT TO R0
     70 004756                                          PUSH    R0              ; SAVE COUNT
        004756  100467                                                                  MOV R0,-(SP)
     71 004757  024616                                  CALL    GETMEM          ; GET SOME MEMORY
     72 004760  100617  000007                          MOV     R0,S.SCHR(R1)   ; STORE POINTER TO IN SUBUNIT PARAMETERS
     73 004762                                          POP     R2              ; RESTORE COUNT IN R2
        004762  104262                                                                  MOV (SP)+,R2
     74 004763  104234                      2$:         MOV     (R3)+,R4        ; MOVE ONE WORD OF S CHAR TO R0
     75 004764  100274                                  MOV     R4,(R0)+        ; MOVE TO SUBUNIT CHAR AREA
     76 004765  117402                                  DEC     R2              ; DECREMENT WORD COUNT
     77 004766  054763                                  BNE     2$              ; IF COUNT UNEXPIRED, BRANCH
     78 004767  104117                      3$:         MOV     (R1),R0         ; GET SUBUNIT PARAMETERS
     79 004770                                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
     80 004770  102207  020000                          BIT     #DCYLS,R0       ; SEE IF DIAGNOSTIC CYLINDERS ARE USED
     81 004772  015002                                  BEQ     4$              ; IF NOT, BRANCH
     82 004773  104202  005521                          MOV     #D$,R2          ; MOVE THE CHARACTER 'D' (TO MAKE DBN) TO R3
     83 004775  104303  001706                          MOV     ST+RBNTRK,R3    ; GET RBN'S PER TRACK
     84 004777  103203  177600                          BIC     #HIBYTE!200,R3  ; CLEAR UNUSED BITS
     85 005001  005005                                  BR      5$              ; BRANCH
     86 005002  104202  005517              4$:         MOV     #L$,R2          ; MOVE THE CHARACTER 'L' (TO MAKE LBN) TO R3
     87 005004  114003                                  CLR     R3              ; FOR LBN'S, NO RBN'S PER TRACK ADDED
     88 005005  100612  000005              5$:         MOV     R2,S.LETR(R1)   ; SAVE
     89 005007  105303  001713                          ADD     ST+LBNTRK,R3    ; ADD LBNS PER TRACK (TO ZERO IF LBN AREA)
     90 005011  103203  177400                          BIC     #HIBYTE,R3      ; CLEAR UNUSED BITS
     91 005013  104030  004504                          MOV     R3,SECTRK       ; SAVE IN SECTORS PER TRACK
     92 005015  100613  000006                          MOV     R3,S.TRKL(R1)   ; SAVE IN TRACK LENGTH
     93 005017  025710                                  CALL    COMPSC          ; COMPUTE SECTORS/GROUP AND SECTORS/CYL
     94 005020  025737                                  CALL    CLCMAX          ; CALCULATE MAXIMUM DBN NUMBER (IN CASE NEEDED)
     95 005021  104117                                  MOV     (R1),R0         ; GET SUBUNIT PARAMETERS
     96 005022                                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
     97 005022  102207  000040                          BIT     #BEUSED,R0      ; SEE IF BEGIN/END SETS ARE USED
     98 005024  015030                                  BEQ     6$              ; IF NOT, BRANCH
     99 005025  102207  000200                          BIT     #ONLYCL,R0      ; SEE IF WE ALLREADY HAVE BEGIN/END SETS
    100 005027  015033                                  BEQ     7$              ; IF SO, BRANCH
    101 005030  025234                      6$:         CALL    CHKCYL          ; CONVERT THE CYLS TO BN'S
    102 005031  115002                                  TST     R2              ; SEE IF AN ERROR OCCURRED
    103 005032  055056                                  BNE     9$              ; IF SO, BRANCH
    104 005033  025411                      7$:         CALL    CHKBES          ; CHECK THE BEGIN/END SETS FOR ERRORS
    105 005034  115002                                  TST     R2              ; SEE IF AN ERROR OCCURRED
    106 005035  055056                                  BNE     9$              ; IF SO, BRANCH
    107 005036  104302  001702                          MOV     ST+LBNCYL,R2    ; R2 CONTAINS LO LBN CYLS
    108 005040  105302  001723                          ADD     ST+XBNCYL,R2    ; ADD LO XBN CYLS TO R2
    109 005042  100612  000002                          MOV     R2,S.SDCL(R1)   ; MOVE TO LO STARTING DIAG CYL
    110 005044  104302  001703                          MOV     ST+LBNCYL+1,R2  ; GET HI ORDER LBN CYLS
    111 005046  045050                                  BCC     8$              ; IF NO CARRY, BRANCH
    112 005047  115402                                  INC     R2              ; PROPOGATE CARRY
    113 005050  100612  000003              8$:         MOV     R2,S.SDCL+1(R1) ; STORE HI STARTING DIAG CYL
    114 005052  114002                                  CLR     R2              ; NO ERRORS
    115 005053  104207  000034                          MOV     #SCHAR1,R0      ; SCHAR1 IS NEXT MODULE
    116 005055  005153                                  BR      15$             ; EXIT
    117 005056  104650  000063  001106  9$:             MOV     U.UNUM(R5),HRQ.03       ; GET STARTING UNIT NUMBER
```

```
                                        ADD     U.SUBU(R5),HRQ.03       ; ADD OFFSET
                                        MOV     HRQ.RQ,R0               ; SET UP TO REPORT ERROR
                                        CALL    HOSTRQ                  ; REPORT
                                        BIS     #DIE,M.PARM             ; FLAG INITIALIZATION ERROR
                                        MOV     #U.SUBP,R1              ; R1 WILL POINT AT SUBUNIT POINTERS
                                        ADD     R5,R1                   ; R1 POINTS AT SUBUNIT POINTERS
                                        ADD     U.SUBU(R5),R1           ; R1 POINTS AT POINTER TO SUBUNIT BEING HANDLED
                                        MOV     (R1),R0                 ; R0 POINTS TO SUBUNIT BEGING HANDLED
                                        MOV     #DROP,R3                ; MOVE DROP FLAG TO R3
                                        MOV     R3,(R0)                 ; DROP THIS SUBUNIT
                                        ASSUME  S.PARM,0
                                        CLR     R2                      ; NO ERRORS
                                10$:    MOV     U.SUBU(R5),R3           ; GET SUBUNIT OFFSET
                                        INC     R3                      ; NEXT SUBUNIT
                                        MOV     R3,U.SUBU(R5)           ; SAVE
                                        CMP     #3,R3                   ; SEE IF VALID OFFSET
                                        BCS     11$                     ; IF NOT, BRANCH
                                                                                        BCC     .+2
                                                                                        BR      11$
                                        ADD     #U.SUBP,R3              ; WILL POINT TO NEXT SUBUNIT POINTER
                                        ADD     R5,R3                   ; ADD POINTER TO UNIT DATABASE
                                        MOV     (R3),R3                 ; GET POINTER
                                        BMI     10$                     ; IF NO SUBUNIT, BRANCH
                                        MOV     (R3),R1                 ; GET SUBUNIT PARAMETERS
                                        ASSUME  S.PARM,0
                                        BMI     10$                     ; IF DROPPED, BRANCH
                                        ASSUME  DROP,100000
                                        BR      12$                     ; EXIT
                                11$:    CLR     R3                      ; NO SUBUNITS
                                        MOV     R3,U.SUBU(R5)           ; SAVE
                                12$:    TST     R3                      ; ANY?
                                        BNE     14$                     ; IF SO, BRANCH
                                        MOV     #GETSER,R0              ; GETSER NEXT MODULE
                                        BR      15$                     ; EXIT
                                13$:    CERROR  5,#SER5                 ; REPORT SECONDARY ERROR
                                                                                        MOV     #SER5,HRQ.05
                                        BIC     #C2DFTL,HRQ.02          ; CHANGE ERROR TO HARD ERROR
                                        MOV     #ERRMC,HRQ.RQ           ; COUNT ERROR
                                        BIS     #DIE,M.PARM             ; FLAG INITIALIZATION ERROR
                                14$:    MOV     #SCHAR0,R0              ; THIS MODULE IS NEXT
                                15$:    CLR     R1                      ; IMMIDATE CALL TO NEXT MODULE
                                        POP     R4                      ; RESTORE R4
                                                                                        MOV (SP)+,R4
                                        BR      JMPRET                  ; RETURN TO SEQNCR
                                        .DSABL  LSB
```

```
                                .SBTTL  TRAV - TRAVERSE THOUGH THE UNITS TO FIND SUBUNIT CHARS THAT MATCH
005156          TRAV:   PUSH    R2              ; SAVE R2 -> UNIT
005156  100462                                                          MOV R2,-(SP)
005157  025167          CALL    SUBTRV          ; GO TRAVERSE SUBUNITS
005160          POP     R2                      ; RESTORE R2
005160  104262                                                          MOV (SP)+,R2
005161  115007          TST     R0              ; FOUND A MATCH
005162  055166          BNE     1$              ; IF SO, BRANCH
005163  104122          MOV     (R2),R2         ; ELSE, CHECK NEXT UNIT
005164  106052          CMP     R5,R2           ; ARE WE POINTING TO SAME UNIT?
005165  055156          BNE     TRAV            ; IF NOT, CONTINUE
005166  000000  1$:     RETURN                  ; ELSE, EXIT

                                .SBTTL  SUBTRV - TRAVERSE THROUGH SUBUNITS TO FIND SUBUNIT CHARS THAT MATCH
005167  104204  000004  SUBTRV: MOV     #4,R4           ; R1 IS COUNTER TO END
005171  115402          INC     R2              ; R2 -> SUBUNIT PARAMETER POINTER
005172                  ASSUME  U.SUBP,1
005172  104223  1$:     MOV     (R2)+,R3        ; R3 HAS ADDRESS OF SUBUNIT PARAMETER
005173  075205          BMI     2$              ; IF MINUS, NOT A GOOD SUBUNIT
005174  104637  000007          MOV     S.SCHR(R3),R0   ; R0 -> SUBUNIT CHAR
005176          PUSH    <R4,R2>
005176  100464                                                          MOV R4,-(SP)
005177  100462                                                          MOV R2,-(SP)
005200  025210          CALL    SCHRCP          ; CHECK SUBUNIT CHARACTERISTICS
005201          POP     <R2,R4>
005201  104262                                                          MOV (SP)+,R2
005202  104264                                                          MOV (SP)+,R4
005203  115007          TST     R0              ; FIND A MATCH?
005204  055207          BNE     3$              ; IF SO, EXIT
005205  117404  2$:     DEC     R4              ; ELSE, ARE ALL SUBUNITS DONE FOR THIS UNIT?
005206  055172          BNE     1$              ; IF NOT, BRANCH
005207  000000  3$:     RETURN

                                .SBTTL  SCHRCP - SUBUNIT CHARACTERISTICS COMPARE
                                ; *** R0 = 0 IF NO MATCH OR R0 -> SUBUNIT CHAR IF THERE IS A MATCH
005210          SCHRCP: PUSH    <R0,R1>
005210  100467                                                          MOV R0,-(SP)
005211  100461                                                          MOV R1,-(SP)
005212  104204  001702          MOV     #ST,R4          ; R4 -> CHARACTERISTICS
005214  104201  000023          MOV     #19.,R1         ; R1 = # OF WORDS TO COMPARE
005216  104242  1$:     MOV     (R4)+,R2        ; COMPARE CHARACTERISTICS
005217  106272          CMP     (R0)+,R2        ; EQUAL?
005220  055230          BNE     2$              ; IF NOT, EXIT
005221  117401          DEC     R1              ; ELSE, CHECK NEXT WORD
005222  055216          BNE     1$              ; IF NOT DONE WITH COMPARISON, BRANCH
005223          POP     <R1,R0>         ; RESTORE REGS
005223  104261                                                          MOV (SP)+,R1
005224  104267                                                          MOV (SP)+,R0
005225  100617  000007          MOV     R0,S.SCHR(R1)   ; AND STORE POINT TO CHAR INTO SUBUNIT PARAM
005227  005233          BR      3$              ; AND EXIT WITH MATCH
005230  2$:     POP     <R1,R0>
005230  104261                                                          MOV (SP)+,R1
005231  104267                                                          MOV (SP)+,R0
005232  114007          CLR     R0              ; R0 = 0 (NO MATCH)
005233  000000  3$:     RETURN
```

```
                                        .SBTTL  CHKCYL - CHECK VALIDITY OF STARTING AND ENDING CYLS, CONVERT TO BEGIN/END SET
005234                                  CHKCYL:
                                        ;
                                        ;       CALCULATE THE STARTING AND ENDING BN'S FROM THE GIVEN STARTING
                                        ;       AND ENDING CYLS
                                        ;
005234                                          PUSH    R5              ; SAVE R4, R5
005234  100465                                                                  MOV R5,-(SP)
005235  104614  000015                        MOV     S.BESS+2(R1),R4 ; R4 HAS LO STARTING CYL
005237  104615  000016                        MOV     S.BESS+3(R1),R5 ; R5 HAS HI STARTING CYL
005241  103205  100000                        BIC     #100000,R5      ; CLEAR EOL FLAG (IF ANY)
005243  025337                              CALL    CYLBN           ; CALCULATE STARTING BN ON THAT CYL
005244  115002                              TST     R2              ; SEE IF ANY ERROR OCCURRED
005245  055335                              BNE     10$             ; IF SO, BRANCH
005246  100617  000015                        MOV     R0,S.BESS+2(R1) ; SAVE LO ORDER STARTING BN
005250  101203  100000                        BIS     #100000,R3      ; SET END-OF-LIST FLAG
005252  100613  000016                        MOV     R3,S.BESS+3(R1) ; SAVE HI ORDER STARTING BN
005254  104615  000014                        MOV     S.BESS+1(R1),R5 ; GET HI ORDER ENDING CYL
005256  106205  177777                        CMP     #-1,R5          ; SEE IF SHOULD DEFAULT TO HIGHEST LBN
005260  015303                              BEQ     4$              ; IF SO, BRANCH
005261  104614  000013                        MOV     S.BESS(R1),R4   ; GET LO ORDER ENDING CYL
005263  105204  000001                        ADD     #1,R4           ; FIND STARTING BN OF NEXT CYL
005265  045267                              BCC     2$              ; IF NO CARRY, BRANCH
005266  115405                              INC     R5              ; PROPOGATE CARRY
005267  025337                          2$: CALL    CYLBN           ; CACULATE STARTING BN OF NEXT CYL
005270  115002                              TST     R2              ; SEE IF ANY ERRORS
005271  055335                              BNE     10$             ; IF SO, BRANCH
005272  107207  000001                        SUB     #1,R0           ; GET LAST BN OF PREVIOUS CYL
005274  045276                              BCC     3$              ; IF NO CARRY, BRANCH
005275  117403                              DEC     R3              ; PROPOGATE CARRY
005276  100617  000013                    3$: MOV     R0,S.BESS(R1)   ; SAVE LO ORDER ENDING BN
005300  100613  000014                        MOV     R3,S.BESS+1(R1) ; SAVE HI ORDER ENDING BN
005302  005334                              BR      9$              ; EXIT
005303  104117                          4$: MOV     (R1),R0         ; GET SUBUNIT PARAMETERS
005304                                      ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
005304  102207  020000                        BIT     #DCYLS,R0       ; SEE IF DIAGNOSTIC CYLINDERS USED
005306  055324                              BNE     6$              ; IF SO, BRANCH
005307  104307  001714                        MOV     ST+LBNHST,R0    ; GET LO ORDER ENDING LBN
005311  107207  000001                        SUB     #1,R0           ; LAST LBN IN LBN AREA
005313  100617  000013                        MOV     R0,S.BESS(R1)   ; SAVE LO ORDER
005315  104307  001715                        MOV     ST+LBNHST+1,R0  ; GET HI ORDER
005317  045321                              BCC     5$              ; IF NO CARRY, BRANCH
005320  117407                              DEC     R0              ; PROPOGATE CARRY
005321  100617  000014                    5$: MOV     R0,S.BESS+1(R1) ; SAVE HI ORDER
005323  005334                              BR      9$              ; EXIT
005324  104307  004502                    6$: MOV     MAXDBN,R0       ; GET LO ORDER MAX DBN
005326  100617  000013                        MOV     R0,S.BESS(R1)   ; SAVE
005330  104307  004503                        MOV     MAXDBN+1,R0     ; GET HI ORDER
005332  100617  000014                        MOV     R0,S.BESS+1(R1) ; SAVE
005334  114002                          9$: CLR     R2              ; NO ERRORS
005335                                  10$:POP     R5              ; RESTORE
005335  104265                                                                  MOV (SP)+,R5
005336  000000                              RETURN
```

```
                                        .SBTTL  CYLBN  - GIVEN A CYLINDER, CALCULATE STARTING BN ON THAT CYLINDER
005337                                  CYLBN:
                                        ;
                                        ;       TAKING THE CYL GIVEN IN R4,R5 CALCULATE THE FIRST BN ON THAT
                                        ;       CYL AND RETURN IN R2,R3.  IF BN EXCEEDS 28 BITS, REPORT
                                        ;       ERROR
                                        ;
005337  114007                          CLR     R0              ; CLEAR THE PRODUCT AREA
005340  114003                          CLR     R3              ; CLEAR THE PRODUCT AREA
005341  115004                          TST     R4              ; SEE IF LO ORDER IS ZERO
005342  055345                          BNE     1$              ; IF NOT, BRANCH
005343  115005                          TST     R5              ; SEE IF HI ORDER IS ZERO
005344  015407                          BEQ     5$              ; IF SO, BRANCH AND EXIT
005345  107204  000001          1$:     SUB     #1,R4           ; ADJUST COUNT
005347  045351                          BCC     2$              ; IF NO BORROW, BRANCH
005350  117405                          DEC     R5              ; PROPOGATE BORROW
005351  105307  004506          2$:     ADD     SECCYL,R0       ; ADD LO SECTORS/CYL TO LO ORDER PROD
005353  045355                          BCC     3$              ; IF NO CARRY, BRANCH
005354  115403                          INC     R3              ; PROPOGATE CARRY
005355  106203  007777          3$:     CMP     #HBHINB,R3      ; SEE IF HI ORDER TOO BIG
005357  045401                          BCC     4$              ; IF NOT, BRANCH
005360                                  DEVFTL  57,#SER20       ; REPORT LBN OVERFLOW
005360  104200  003644  001107                                                  MOV     #ER57,HRQ.04
005363  104200  004047  001110                                                  MOV     #SER20,HRQ.05
005366  104202  047731                                                          MOV     #57!FTLDEV+4000.,R2
005370  104020  001105                                                          MOV     R2,HRQ.02
005372  104200  005372  001104                                                  MOV     #.,HRQ.01
005375  104200  060014  001103                                                  MOV     #ERRMC,HRQ.RQ
005400  005410                          BR      6$              ; EXIT
005401  107204  000001          4$:     SUB     #1,R4           ; DECREMENT LO ORDER COUNT
005403  045351                          BCC     2$              ; IF INCOMPLETE, BRANCH
005404  107205  000001                  SUB     #1,R5           ; DECREMENT HI ORDER COUNT
005406  045351                          BCC     2$              ; IF INCOMPLETE, BRANCH
005407  114002                  5$:     CLR     R2              ; NO ERRORS
005410  000000                  6$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  CHKBES - CHECK THE BEGIN/END SETS FOR ERRORS
005411                          CHKBES:
                                ;
                                ;       CHECK THAT THE BEGIN BLOCK <= END BLOCK IN EACH BEGIN/END SET,
                                ;       THAT THE END BLOCK < THE BEGIN BLOCK OF THE NEXT BEGIN/END SET,
                                ;       AND THAT THE LAST END BLOCK IS A VALID LBN OR DBN FOR THAT DRIVE.
                                ;
005411                          PUSH    R5
005411  100465                                                                  MOV R5,-(SP)
005412  104203  000013         MOV     #S.BESS,R3      ; R3 WILL POINT TO ENDING BLOCK NUMBER
005414  105013                 ADD     R1,R3           ; R3 POINTS TO ENDING BN
005415  104202  000015         MOV     #S.BESS+2,R2    ; R2 WILL POINT TO STARTING BN
005417  105012                 ADD     R1,R2           ; R2 POINTS TO STARTING BN
005420  024623                 1$:     CALL    CBB2            ; COMPARE
005421  045473                 BCC     2$              ; IF NO ERROR, BRANCH
005422  104112                 MOV     (R1),R2         ; GET SUBUNIT PARAMETERS
005423                         ASSUME  S.PARM,0
005423  102202  000040         BIT     #BEUSED,R2      ; SEE IF BEGIN/END SETS USED
005425  015431                 BEQ     20$             ; IF NOT, BRANCH
005426  102202  000200         BIT     #ONLYCL,R2      ; SEE IF ONLY THE CYLINDER IS SPECIFIED
005430  015452                 BEQ     21$             ; IF NOT, BRANCH
005431                         20$:    DEVFTL  55,#SER20       ; REPORT BEGIN CYL > ENDING CYL
005431  104200  003552  001107                                                  MOV     #ER55,HRQ.04
005434  104200  004047  001110                                                  MOV     #SER20,HRQ.05
005437  104202  047727                                                          MOV     #55!FTLDEV+4000.,R2
005441  104020  001105                                                          MOV     R2,HRQ.02
005443  104200  005443  001104                                                  MOV     #.,HRQ.01
005446  104200  060014  001103                                                  MOV     #ERRMC,HRQ.RQ
005451  005706                 BR      7$              ; BRANCH TO ERROR EXIT
005452                         21$:    DEVFTL  50,#SER20       ; REPORT BEGIN > ENDING BN
005452  104200  003315  001107                                                  MOV     #ER50,HRQ.04
005455  104200  004047  001110                                                  MOV     #SER20,HRQ.05
005460  104202  047722                                                          MOV     #50!FTLDEV+4000.,R2
005462  104020  001105                                                          MOV     R2,HRQ.02
005464  104200  005464  001104                                                  MOV     #.,HRQ.01
005467  104200  060014  001103                                                  MOV     #ERRMC,HRQ.RQ
005472  005706                 BR      7$              ; BRANCH TO ERROR EXIT
005473  104625  000001         2$:     MOV     1(R2),R5        ; SEE IF THE END-OF-LIST HAS BEEN FOUND
005475  075527                 BMI     4$              ; IF SO, BRANCH
005476  105202  000004         ADD     #4,R2           ; POINT TO NEXT BEGIN SET
005500  024623                 CALL    CBB2            ; COMPARE
005501                         BCS     3$              ; BRANCH
005501  045503                                                                  BCC     .+2
005502  005524                                                                  BR      3$
005503                         DEVFTL  51,#SER20       ; STARTING BN <= PREVIOUS ENDING BN
005503  104200  003362  001107                                                  MOV     #ER51,HRQ.04
005506  104200  004047  001110                                                  MOV     #SER20,HRQ.05
005511  104202  047723                                                          MOV     #51!FTLDEV+4000.,R2
005513  104020  001105                                                          MOV     R2,HRQ.02
005515  104200  005515  001104                                                  MOV     #.,HRQ.01
005520  104200  060014  001103                                                  MOV     #ERRMC,HRQ.RQ
005523  005706                 BR      7$              ; ERROR EXIT
005524  105203  000004         3$:     ADD     #4,R3           ; R3 NOW POINTS TO NEXT END SET
005526  005420                 BR      1$              ; LOOP
005527  104112                 4$:     MOV     (R1),R2         ; GET SUBUNIT CHARACTERISTICS
005530                         ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
005530  102202  020000         BIT     #DCYLS,R2       ; SEE IF DIAGNOSTIC CYLINDERS ARE IN USE
```

```
     37 005532  055613                          BNE     5$              ; IF SO, BRANCH
     38 005533  104202  001714                  MOV     #ST+LBNHST,R2   ; R2 POINTS TO LBNS IN HOST AREA
     39 005535  024623                          CALL    CBB2            ; COMPARE
     40 005536                                  BCS     6$              ; EXIT
        005536  045540                                                                  BCC     .+2
        005537  005705                                                                  BR      6$
     41 005540  104112                          MOV     (R1),R2         ; GET SUBUNIT CHARACTERISTICS
     42 005541                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
     43 005541  102202  000040                  BIT     #BEUSED,R2      ; SEE IF TRACKS/GROUPS IN USE
     44 005543  015547                          BEQ     10$             ; IF SO, BRANCH
     45 005544  102202  000200                  BIT     #ONLYCL,R2      ; SEE IF B/E SET COMPUTED BY CYLINDERS
     46 005546  015556                          BEQ     11$             ; IF NOT, BRANCH
     47 005547                          10$:    CERROR  6,#L$           ; LBN ERROR
        005547  104200  005517  001111                                                  MOV     #L$,HRQ.06
     48 005552                                  CERROR  7,#L$           ; LBN ERROR
        005552  104200  005517  001112                                                  MOV     #L$,HRQ.07
     49 005555  005635                          BR      14$             ; EXIT
     50 005556  107200  000001  001714  11$:    SUB     #1,ST+LBNHST    ; DECREMENT LO LBN COUNT BY 1
     51 005561  045564                          BCC     9$              ; IF NO BORROW, BRANCH
     52 005562  117400  001715                  DEC     ST+LBNHST+1     ; PROPOGATE BORROW
     53 005564                          9$:     DEVFTL  52,<#SER20,ST+LBNHST,ST+LBNHST+1> ; LAST ENDING BN > MAX
        005564  104200  003405  001107                                                  MOV     #ER52,HRQ.04
        005567  104200  004047  001110                                                  MOV     #SER20,HRQ.05
        005572  104300  001714  001111                                                  MOV     ST+LBNHST,HRQ.06
        005575  104300  001715  001112                                                  MOV     ST+LBNHST+1,HRQ.07
        005600  104202  047724                                                          MOV     #52!FTLDEV+4000.,R2
        005602  104020  001105                                                          MOV     R2,HRQ.02
        005604  104200  005604  001104                                                  MOV     #.,HRQ.01
        005607  104200  060014  001103                                                  MOV     #ERRMC,HRQ.RQ
     54 005612  005706                          BR      7$              ; ERROR EXIT
     55 005613  104032                  5$:     MOV     R3,R2           ; R2 POINTS TO LAST END SET
     56 005614  104203  004502                  MOV     #MAXDBN,R3      ; R3 POINTS TO MAXIMUM DBN
     57 005616  024623                          CALL    CBB2            ; COMPARE
     58 005617  045705                          BCC     6$              ; EXIT
     59 005620  104112                          MOV     (R1),R2         ; GET SUBUNIT CHARACTERISTICS
     60 005621                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
     61 005621  102202  000040                  BIT     #BEUSED,R2      ; SEE IF TRACKS/GROUPS IN USE
     62 005623  015627                          BEQ     12$             ; IF SO, BRANCH
     63 005624  102202  000200                  BIT     #ONLYCL,R2      ; SEE IF B/E SET COMPUTED BY CYLINDERS
     64 005626  015656                          BEQ     13$             ; IF NOT, BRANCH
     65 005627                          12$:    CERROR  6,#D$           ; DBN ERROR
        005627  104200  005521  001111                                                  MOV     #D$,HRQ.0o
     66 005632                                  CERROR  7,#D$           ; DBN ERROR
        005632  104200  005521  001112                                                  MOV     #D$,HRQ.07
     67 005635                          14$:    DEVFTL  62,#SER20       ; REPORT ERROR
        005635  104200  003765  001107                                                  MOV     #ER62,HRQ.04
        005640  104200  004047  001110                                                  MOV     #SER20,HRQ.05
        005643  104202  047736                                                          MOV     #62!FTLDEV+4000.,R2
        005645  104020  001105                                                          MOV     R2,HRQ.02
        005647  104200  005647  001104                                                  MOV     #.,HRQ.01
        005652  104200  060014  001103                                                  MOV     #ERRMC,HRQ.RQ
     68 005655  005706                          BR      7$              ; EXIT
     69 005656                          13$:    DEVFTL  52,<#SER20,MAXDBN,MAXDBN+1> ; LAST ENDING BN > MAX
        005656  104200  003405  001107                                                  MOV     #ER52,HRQ.04
        005661  104200  004047  001110                                                  MOV     #SER20,HRQ.05
        005664  104300  004502  001111                                                  MOV     MAXDBN,HRQ.06
        005667  104300  004503  001112                                                  MOV     MAXDBN+1,HRQ.07
```

```
        005672  104202  047724                                          MOV     #52!FTLDEV+4000.,R2
        005674  104020  001105                                          MOV     R2,HRQ.02
        005676  104200  005676  001104                                  MOV     #.,HRQ.01
        005701  104200  060014  001103                                  MOV     #ERRMC,HRQ.RQ
    70  005704  005706                      BR      7$          ; ERROR EXIT
    71  005705  114002              6$:     CLR     R2          ; FLAG AS NO ERROR
    72  005706                      7$:     POP     R5          ; RESTORE REGISTER
        005706  104265                                                  MOV (SP)+,R5
    73  005707  000000                      RETURN              ; RETURN TO INSCHR
```

```
                                        .SBTTL  COMPSC - COMPUTE SECTORS/GROUP AND SECTORS/CYLINDER
005710                                  COMPSC:
                                        ;
                                        ;       COMPUTE HOW MANY SECTORS/GROUP AND SECTORS/CYLINDER
                                        ;       GIVEN HOW MANY SECTORS/TRACK IN SECTRK
                                        ;
005710  114007                                  CLR     R0                      ; CLEAR RUNNING TOTAL
005711  104302  001705                          MOV     ST+TRKGRP,R2            ; GET NUMBER OF TRACKS PER GROUP
005713  103202  177400                          BIC     #HIBYTE,R2              ; CLEAR UNUSED BITS
005715  105307  004504                  1$:     ADD     SECTRK,R0               ; ADD NUMBER OF SECTORS/TRACK TO RUNNING TOTAL
005717  117402                                  DEC     R2                      ; DECREMENT COUNT
005720  055715                                  BNE     1$                      ; IF COUNT INCOMPLETE, BRANCH
005721  104070  004505                          MOV     R0,SECGRP               ; SAVE SECTORS/GROUP
005723  114007                                  CLR     R0                      ; CLEAR RUNNING TOTAL
005724  104302  001704                          MOV     ST+GRPCYL,R2            ; GET GROUPS/CYLINDER
005726  103202  177400                          BIC     #HIBYTE,R2              ; CLEAR UNUSED BITS
005730  105307  004505                  2$:     ADD     SECGRP,R0               ; ADD SECTORS/GROUP TO RUNNING TOTAL
005732  117402                                  DEC     R2                      ; DECREMENT COUNT
005733  055730                                  BNE     2$                      ; IF COUNT INCOMPLETE, BRANCH
005734  104070  004506                          MOV     R0,SECCYL               ; STORE SECTORS/CYLINDER
005736  000000                                  RETURN                          ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  CLCMAX - CALCULATE THE MAXIMUM WRITEABLE DBN
005737                                  CLCMAX:
                                        ;
                                        ;       CALCULATE THE MAXIMUM DBN NUMBER ALLOWED, AND PLACE IT IN MAXDBN
                                        ;       FOR BEGIN/END SET GENERATION AND TESTING (BAD BLOCKS TOO)
                                        ;
005737                                          PUSH    R1              ; SAVE REGISTER
005737  100461                                                                  MOV R1,-(SP)
005740  114007                                  CLR     R0              ; CLEAR PRODUCT AREA
005741  114001                                  CLR     R1              ; CLEAR PRODUCT AREA
005742  104302  001724                         MOV     ST+DBNCYL,R2    ; GET DIAG CYL COUNT
005744  110702                                  SWAB    R2              ; MOVE TO LO ORDER BYTE
005745  103202  177400                         BIC     #HIBYTE,R2      ; CLEAR UNUSED BITS
005747  105307  004506          1$:             ADD     SECCYL,R0       ; ADD SECTORS/CYL TO TOTAL
005751  045753                                  BCC     2$              ; IF NO CARRY, BRANCH
005752  115401                                  INC     R1              ; INCREMENT HI WORD
005753  117402                  2$:             DEC     R2              ; DECREMENT COUNT
005754  055747                                  BNE     1$              ; IF NO CARRY, BRANCH
005755  104302  001724                         MOV     ST+DBNCYL,R2    ; GET NUMBER OF READ ONLY GROUPS
005757  103202  177400                         BIC     #HIBYTE,R2      ; CLEAR UNUSED BITS
005761  107307  004505          3$:             SUB     SECGRP,R0       ; SUBTRACT NUMBER OF SECTORS/GROUP
005763  045765                                  BCC     4$              ; IF NO BORROW, BRANCH
005764  117401                                  DEC     R1              ; PROPOGATE BORROW
005765  117402                  4$:             DEC     R2              ; DECREMENT COUNT
005766  055761                                  BNE     3$              ; IF INCOMPLETE, BRANCH
005767  107207  000001                         SUB     #1,R0           ; ADJUST FOR DBNS STARTING AT ZERO
005771  045773                                  BCC     5$              ; IF NO BORROW, BRANCH
005772  117401                                  DEC     R1              ; PROPOGATE BORROW
005773  104070  004502          5$:             MOV     R0,MAXDBN       ; SAVE LO ORDER WORD
005775  104010  004503                         MOV     R1,MAXDBN+1     ; SAVE HI ORDER WORD
005777                                          POP     R1              ; RESTORE REGISTERS
005777  104261                                                                  MOV (SP)+,R1
006000  000000                                  RETURN                  ; RETURN TO INSCHR
                                                .IF     LE,MAXADR-.
                                MAXADR  =       .+1
                                                .ENDC
```

```
********** OVERLAY MODULE SCHAR1 - SET UP SUBUNITS, CHECK TRAC

                                        .SBTTL  ********** OVERLAY MODULE SCHAR1 - SET UP SUBUNITS, CHECK TRACK/GROUP PARAMETERS
 006001                                         DMOVLY  S1,AREAI
 006001  000105                                .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;
                                        ;
                                        ;
 000034                                 SCHAR1  =       SCHAR0+1
                                        ;
                                        ;       SCHAR1 WILL INITILIZE THE SUBUNIT PARAMETERS (TRACK/GROUP ONLY)
                                        ;
                                        ;
                                        ;
                                        .ENABL  LSB
 004701                                 PUSH    R4              ; SAVE SUBUNIT POINTER
 004701  100464                                                                  MOV R4,-(SP)
 004702  104141                        MOV     (R4),R1         ; GET SUBUNIT PARAMETERS
 004703                                 ASSUME  S.PARM,0
 004703  102201  004000                BIT     #RONLY,R1       ; SEE IF READ ONLY
 004705  014753                        BEQ     5$              ; IF NOT, BRANCH
 004706  102201  002000                BIT     #WONLY,R1       ; SEE IF WRITE ONLY IS SET TOO
 004710  014727                        BEQ     6$              ; IF NOT, BRANCH
 004711                                 DEVFTL  28              ; SETUP ERROR
 004711  104200  002415  001107                                          MOV     #ER28,HRQ.04
 004714  104202  047674                                                  MOV     #28!FTLDEV+4000.,R2
 004716  104020  001105                                                  MOV     R2,HRQ.02
 004720  104200  004720  001104                                          MOV     #.,HRQ.01
 004723  104200  060014  001103                                          MOV     #ERRMC,HRQ.RQ
 004726  004766                        BR      4$              ; BRANCH TO REPORT
 004727  102201  040000        6$:     BIT     #INITW,R1       ; SEE IF INITIAL WRITE
 004731  014753                        BEQ     5$              ; IF NOT, BRANCH
 004732  103201  040000                BIC     #INITW,R1       ; NO INITIAL WRITE
 004734  100141                        MOV     R1,(R4)         ; SAVE PARAMETERS
 004735                                 ASSUME  S.PARM,0
 004735                                 MSSG    4               ; REPORT NO INITIAL WRITE
 004735  104200  005701  001105                                          MOV     #MS4,HRQ.02
 004740  100467                                                                  MOV R0,-(SP)
 004741  104650  000063  001104                                          MOV     U.UNUM(R5),HRQ.01
 004744  105650  000050  001104                                          ADD     U.SUBU(R5),HRQ.01
 004747  104207  060015                                                  MOV     #MESSAG,R0
 004751  021053                                                          CALL    HOSTRQ
 004752  104267                                                                  MOV (SP)+,R0
 004753                         5$:
                                        ;       BIT     #DCYLS,R1       ; SEE IF IN DIAGNOSTIC AREA
                                        ;       BEQ     7$              ; IF NOT, BRANCH
                                        ;       MSSG    5               ; REPORT ASSUMPTION
                                        ;7$:
 004753  104041                        MOV     R4,R1           ; R1 NOW POINTS TO SUBUNIT DATASTRUCTURE
 004754  025052                        CALL    CHKBB           ; CHECK BAD BLOCKS
 004755  115002                        TST     R2              ; ANY ERRORS ?
 004756  054766                        BNE     4$              ; IF SO, BRANCH
 004757  104117                        MOV     (R1),R0         ; GET SUBUNIT CHARACTERISTICS
 004760                                 ASSUME  S.PARM,0
 004760  102207  000040                BIT     #BEUSED,R0      ; SEE IF BEGIN/END SETS WERE USED
 004762  055014                        BNE     1$              ; IF SO, SKIP EVERYTHING
```

```
     61 004763  025223                          CALL    CHKTG           ; COMPUTE TRACK/GROUP DATABASE
     62 004764  115002                          TST     R2              ; SEE IF ANY ERRORS
     63 004765  015014                          BEQ     1$              ; IF NOT, BRANCH
     64 004766  104650  000063  001106  4$:     MOV     U.UNUM(R5),HRQ.03       ; GET STARTING UNIT NUMBER
     65 004771  105650  000050  001106          ADD     U.SUBU(R5),HRQ.03       ; ADD OFFSET
     66 004774  104307  001103                  MOV     HRQ.RQ,R0       ; SET UP TO REPORT ERROR
     67 004776  021053                          CALL    HOSTRQ          ; REPORT
     68 004777  101200  000002  002223          BIS     #DIE,M.PARM     ; FLAG INITIALIZATION ERROR
     69 005002  104201  000C01                  MOV     #U.SUBP,R1      ; R1 WILL POINT AT SUBUNIT POINTERS
     70 005004  105051                          ADD     R5,R1           ; R1 POINTS AT SUBUNIT POINTERS
     71 005005  105651  000050                  ADD     U.SUBU(R5),R1   ; R1 POINTS AT POINTER TO SUBUNIT BEING HANDLED
     72 005007  104117                          MOV     (R1),R0         ; R0 POINTS TO SUBUNIT BEGING HANDLED
     73 005010  104203  100000                  MOV     #DROP,R3        ; MOVE DROP FLAG TO R3
     74 005012  100173                          MOV     R3,(R0)         ; DROP THIS SUBUNIT
     75 005013                                  ASSUME  S.PARM,0
     76 005013  114002                          CLR     R2              ; NO ERRORS
     77 005014  104653  000050          1$:     MOV     U.SUBU(R5),R3   ; GET SUBUNIT OFFSET
     78 005016  115403                          INC     R3              ; NEXT SUBUNIT
     79 005017  100653  000050                  MOV     R3,U.SUBU(R5)   ; SAVE
     80 005021  106203  000003                  CMP     #3,R3           ; SEE IF VALID OFFSET
     81 005023                                  BCS     23$             ; IF NOT, BRANCH
        005023  045025                                                                  BCC     .+2
        005024  005035                                                                  BR      23$
     82 005025  105203  000001                  ADD     #U.SUBP,R3      ; WILL POINT TO NEXT SUBUNIT POINTER
     83 005027  105053                          ADD     R5,R3           ; ADD POINTER TO UNIT DATABASE
     84 005030  104133                          MOV     (R3),R3         ; GET POINTER
     85 005031  075014                          BMI     1$              ; IF NO SUBUNIT, BRANCH
     86 005032  104131                          MOV     (R3),R1         ; GET SUBUNIT PARAMETERS
     87 005033                                  ASSUME  S.PARM,0
     88 005033  075014                          BMI     1$              ; IF DROPPED, BRANCH
     89 005034                                  ASSUME  DROP,100000
     90 005034  005040                          BR      22$             ; EXIT
     91 005035  114003                  23$:    CLR     R3              ; NO SUBUNITS
     92 005036  100653  000050                  MOV     R3,U.SUBU(R5)   ; SAVE
     93 005040  115003                  22$:    TST     R3              ; ANY?
     94 005041  055045                          BNE     2$              ; IF SO, BRANCH
     95 005042  104207  000035                  MOV     #GETSER,R0      ; GETSER NEXT MODULE
     96 005044  005047                          BR      3$              ; EXIT
     97 005045  104207  000033          2$:     MOV     #SCHAR0,R0      ; GO BACK TO FIRST CHARACTERISTICS MODULE
     98 005047  114001                  3$:     CLR     R1              ; IMMIDATE CALL
     99 005050                                  POP     R4              ; RESTORE
        005050  104264                                                                  MOV (SP)+,R4
    100 005051  003231                          BR      JMPRET          ; RETURN TO SEQUENCER
    101                                         .DSABL  LSB
```

```
                                .SBTTL  CHKBB  - CHECK THE BAD BLOCKS FOR ERRORS
005052                          CHKBB:
                                ;
                                ;       CHECK THE BAD BLOCKS TO ASSURE THEY DON'T EXCEED THE MAXIMUM BLOCK
                                ;       NUMBER
                                ;
005052                          PUSH    R5              ; SAVE THE REGISTERS
005052  100465                                                          MOV R5,-(SP)
005053  104613  000012          MOV     S.BADP(R1),R3   ; GET POINTER TO BAD BLOCKS
005055  015220                  BEQ     5$              ; IF NO BAD BLOCKS, BRANCH
005056  104632  000001  1$:     MOV     1(R3),R2        ; IS THIS THE LAST BAD BLOCK?
005060  075112                  BMI     2$              ; IF SO, BRANCH
005061  104032                  MOV     R3,R2           ; R2 WILL POINT TO NEXT BAD BLOCK
005062  105202  000002          ADD     #2,R2           ; R2 POINTS TO NEXT BAD BLOCK
005064  024623                  CALL    CBB2            ; COMPARE
005065  045071                  BCC     7$              ; IF ERROR, BRANCH
005066  105203  000002          ADD     #2,R3           ; POINT TO NEXT BAD BLOCK
005070  005056                  BR      1$              ; LOOP
005071                  7$:     DEVFTL  53,#SER20       ; DUPLICATE BAD BLOCKS
005071  104200  003465  001107                                          MOV     #ER53,HRQ.04
005074  104200  004047  001110                                          MOV     #SER20,HRQ.05
005077  104202  047725                                                  MOV     #53!FTLDEV+4000.,R2
005101  104020  001105                                                  MOV     R2,HRQ.02
005103  104200  005103  001104                                          MOV     #.,HRQ.01
005106  104200  060014  001103                                          MOV     #ERRMC,HRQ.RQ
005111  005221                  BR      6$              ; BRANCH
005112  104117          2$:     MOV     (R1),R0         ; GET SUBUNIT PARAMETERS
005113                          ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
005113  102207  020000          BIT     #DCYLS,R0       ; SEE IF DIAGNOSTIC CYLINDERS IN USE
005115  055164                  BNE     3$              ; IF SO, BRANCH
005116  104612  000007          MOV     S.SCHR(R1),R2   ; R2 POINTS TO SUBUNIT CHARACTERISTICS
005120  105202  000012          ADD     #LBNHST,R2      ; R2 POINTS TO NUMBER OF LBNS IN HOST AREA
005122  024623                  CALL    CBB2            ; COMPARE
005123  045125                  BCC     8$              ; IF ERROR, BRANCH
005124  005220                  BR      5$              ; EXIT WITH NO ERROR
005125  104125          8$:     MOV     (R2),R5         ; GET LO ORDER MAX LBN
005126  107205  000001          SUB     #1,R5           ; DECREMENT COUNT
005130  100125                  MOV     R5,(R2)         ; SAVE
005131  045137                  BCC     9$              ; IF NO CARRY, BRANCH
005132  104625  000001          MOV     1(R2),R5        ; GET HI ORDER
005134  117405                  DEC     R5              ; PROPOGATE CARRY
005135  100625  000001          MOV     R5,1(R2)        ; SAVE
005137                  9$:     DEVFTL  54,<#SER20,(R2)+,(R2)>  ; BAD BLOCK > MAXIMUM
005137  104200  003502  001107                                          MOV     #ER54,HRQ.04
005142  104200  004047  001110                                          MOV     #SER20,HRQ.05
005145  104220  001111                                                  MOV     (R2)+,HRQ.06
005147  104120  001112                                                  MOV     (R2),HRQ.07
005151  104202  047726                                                  MOV     #54!FTLDEV+4000.,R2
005153  104020  001105                                                  MOV     R2,HRQ.02
005155  104200  005155  001104                                          MOV     #.,HRQ.01
005160  104200  060014  001103                                          MOV     #ERRMC,HRQ.RQ
005163  005221                  BR      6$              ; ERROR EXIT
005164  104032          3$:     MOV     R3,R2           ; R2 POINTS TO LAS BAD BLOCK
005165  104203  004502          MOV     #MAXDBN,R3      ; R3 POINTS TO MAXIMUM DBN
005167  024623                  CALL    CBB2            ; COMPARE
005170  045220                  BCC     5$              ; IF NO ERROR, BRANCH
005171                  4$:     DEVFTL  54,<#SER20,MAXDBN,MAXDBN+1> ; BAD BLOCK > MAXIMUM
```

```
        005171  104200  003502  001107                                  MOV     #ER54,HRQ.04
        005174  104200  004047  001110                                  MOV     #SER20,HRQ.05
        005177  104300  004502  001111                                  MOV     MAXDBN,HRQ.06
        005202  104300  004503  001112                                  MOV     MAXDBN+1,HRQ.07
        005205  104202  047726                                          MOV     #54!FTLDEV+4000.,R2
        005207  104020  001105                                          MOV     R2,HRQ.02
        005211  104200  005211  001104                                  MOV     #.,HRQ.01
        005214  104200  060014  001103                                  MOV     #ERRMC,HRQ.RQ
    43  005217  005221                      BR      6$          ; BRANCH TO EXIT
    44  005220  114002              5$:     CLR     R2          ; FLAG AS NO ERRORS
    45  005221                      6$:     POP     R5          ; RESTORE THE REGISTERES
        005221  104265                                                  MOV (SP)+,R5
    46  005222  000000                      RETURN              ; RETURN TO INSCHR
```

```
                                        .SBTTL  CHKTG - CHECK THE TRACK/GROUPS FOR ERRORS, AND CONVERT TO BN'S
005223                                  CHKTG:
                                        ;
                                        ;       CONVERT THE TRACKS/GROUPS TO SECTOR COUNTS, AND FIND THE MAXIMUM
                                        ;       NUMBER OF 'LOOPS' THAT SETUP CAN RUN THROUGH TO TEST ALL
                                        ;       TRACKS ON THE TESTED CYLINDERS ONLY
                                        ;
                                        ;
                                        ;       FIRST COPY THE TRACKS/GROUPS TO A TEMP LOCATION
                                        ;
005223                                  PUSH    R5                      ; SAVE R5
005223  100465                                                                  MOV R5,-(SP)
005224  104207  004507                  MOV     #TGS,R0                 ; POINT TO TEMPORARY STORAGE LOCATION
005226  104202  000017                  MOV     #S.TGSS,R2              ; R2 WILL POINT TO T/G LIST
005230  105012                          ADD     R1,R2                   ; R2 POINTS TO T/G LIST
005231  104225                  1$:     MOV     (R2)+,R5                ; GET WORD
005232  100275                          MOV     R5,(R0)+                ; SAVE IN TEMP LOCATION
005233  035231                          BPL     1$                      ; COPY ENTIRE LIST
                                        ;
                                        ;       NOW MOVE THE STARTING CYLINDER TO THE INITIAL OFFSET AREA
                                        ;
005234  104612  000016                  MOV     S.TGOF+1(R1),R2         ; GET HI ORDER STARTING CYL
005236  103202  170000                  BIC     #^CHBHINB,R2            ; CLEAR UNUSED BITS
005240  100612  000016                  MOV     R2,S.TGOF+1(R1)         ; SAVE
                                        ;
                                        ;       CHECK THAT THE MAXIMUM TRACK/GROUP IS VALID
                                        ;
005242  104117                          MOV     (R1),R0                 ; GET SUBUNIT PARAMETERS
005243                                  ASSUME  S.PARM,0                ; ASSUME THAT S.PARM IS ZERO
005243  104642  000007                  MOV     S.SCHR(R4),R2           ; R2 POINTS TO SUBUNIT CHAR
005245  102207  000020                  BIT     #TRACKS,R0              ; SEE IF PROCESSING GROUPS OR TRACKS
005247  055256                          BNE     2$                      ; IF TRACKS, BRANCH
005250  104627  000002                  MOV     GRPCYL(R2),R0           ; GET GROUPS PER CYLINDER
005252  104200  005513  005505          MOV     #GRP$,TG$               ; PRINT 'GROUP' IF ERROR
005255  005263                          BR      3$                      ; BRANCH
005256  104627  000003          2$:     MOV     TRKGRP(R2),R0           ; GET TRACKS PER GROUP
005260  104200  005507  005505          MOV     #TRK$,TG$               ; PRINT 'TRACK' IF ERROR
005263  103207  177400          3$:     BIC     #HIBYTE,R0              ; CLEAR UNUSED BITS
005265  104202  004507                  MOV     #TGS,R2                 ; POINT TO TRACK/GROUP LIST
005267  104223                  4$:     MOV     (R2)+,R3                ; SEE IF THIS WORD IS END-OF-LIST
005270  035267                          BPL     4$                      ; IF NOT, BRANCH
005271  103203  177400                  BIC     #HIBYTE,R3              ; CLEAR EOL FLAG
005273  106037                          CMP     R3,R0                   ; SEE IF WITHIN LIMITS
005274  075324                          BMI     5$                      ; IF SO, BRANCH
005275  117407                          DEC     R0                      ; ADJUST T/G MAX FOR 0-N
005276                                  DEVFTL  58,<#SER20,TG$,R0>      ; REPORT ERROR
005276  104200  005717  001107                          MOV     #ER58,HRQ.04
005301  104200  004047  001110                          MOV     #SER20,HRQ.05
005304  104300  005505  001111                          MOV     TG$,HRQ.06
005307  104070  001112                                  MOV     R0,HRQ.07
005311  104202  047732                                  MOV     #58!FTLDEV+4000.,R2
005313  104020  001105                                  MOV     R2,HRQ.02
005315  104200  005315  001104                          MOV     #.,HRQ.01
005320  104200  060014  001103                          MOV     #ERRMC,HRQ.RQ
005323  005503                          BR      17$                     ; EXIT
```

```
                                        ;
                                        ;       NOW COMPUTE THE NUMBER OF SECTORS FROM THE START OF THE CYLINDER/GROUP
                                        ;       TO THE FIRST TRACK/GROUP TO TEST
                                        ;
005324  104207  004507               5$:      MOV     #TGS,R0                 ; R0 POINTS TO TRACK/GROUP LIST
005326  104203  000015                       MOV     #S.TGOF,R3              ; R3 WILL POINT TO T/G INITIAL OFFSET
005330  105013                               ADD     R1,R3                   ; R3 POINTS TO T/G INITIAL OFFSET
005331  104172                               MOV     (R0),R2                 ; GET FIRST T/G
005332  103202  177400                       BIC     #HIBYTE,R2              ; CLEAR UNUSED BITS
005334  025506                               CALL    COMPDP                  ; COMPUTE SECTORS FROM CYL TO INIT T/G
005335  105135                               ADD     (R3),R5                 ; ADD TO INITIAL OFFSET
005336  100235                               MOV     R5,(R3)+                ; MOVE BACK
005337  045343                               BCC     6$                      ; IF NO CARRY, BRANCH
005340  104135                               MOV     (R3),R5                 ; GET HI ORDER WORD
005341  115405                               INC     R5                      ; PROPOGATE CARRY
005342  100135                               MOV     R5,(R3)                 ; SAVE
005343  115403                         6$:   INC     R3                      ; POINT TO NEXT AREA
                                        ;
                                        ;       NOW COMPUE HOW MANY SECTORS FROM THE LAST TRACK/GROUP TO TEST TO THE
                                        ;       NEXT TRACK/GROUP TO TEST
                                        ;
005344  104174                         7$:   MOV     (R0),R4                 ; GET TRACK/GROUP
005345  075404                               BMI     8$                      ; IF EOL, BRANCH
005346  104672  000001                       MOV     1(R0),R2                ; GET NEXT TRACK
005350  103202  177400                       BIC     #HIBYTE,R2              ; CLEAR EOL FLAG, IF ANY
005352  107272                               SUB     (R0)+,R2                ; GET HOW MANY T/G'S BETWEEN LAST/NEXT
005353  015355                               BEQ     20$                     ; IF ZERO, ERROR
005354  035401                               BPL     18$                     ; SHOULD BE AT LEAST 1 T/G BETWEEN
005355                                 20$:  DEVFTL  59,<#SER20,TG$>         ; REPORT ERROR
005355  104200  003750  001107                                                       MOV     #ER59,HRQ.04
005360  104200  004047  001110                                                       MOV     #SER20,HRQ.05
005363  104300  005505  001111                                                       MOV     TG$,HRQ.06
005366  104202  047733                                                             MOV     #59!FTLDEV+4000.,R2
005370  104020  001105                                                             MOV     R2,HRQ.02
005372  104200  005372  001104                                                     MOV     #.,HRQ.01
005375  104200  060014  001103                                                     MOV     #ERRMC,HRQ.RQ
005400  005503                               BR      17$                     ; EXIT
005401  025506                         18$:  CALL    COMPDP                  ; COMPUTE HOW MANY SECTORS TO NEXT T/G
005402  100235                               MOV     R5,(R3)+                ; STORE THE OFFSET
005403  005344                               BR      7$                      ; LOOP
                                        ;
                                        ;       NOW COMPUTE HOW MANY SECTORS FROM THE LAST T/G TO TEST TO THE FIRST
                                        ;
005404  104115                         8$:   MOV     (R1),R5                 ; GET SUBUNIT PARAMETERS
005405                                       ASSUME  S.PARM,0                ; ASSUME THAT S.PARM IS ZERO
005405  104612  000007                       MOV     S.SCHR(R1),R2           ; R2 POINTS TO SUBUNIT CHAR
005407  102205  000020                       BIT     #TRACKS,R5              ; SEE IF USING GROUPS OR TRACKS
005411  055415                               BNE     9$                      ; IF TRACKS, BRANCH
005412  104622  000002                       MOV     GRPCYL(R2),R2           ; GET GROUPS/CYL
005414  005417                               BR      10$                     ; BRANCH
005415  104622  000003                 9$:   MOV     TRKGRP(R2),R2           ; GET TRACKS/GROUP
005417  105302  004507                 10$:  ADD     TGS,R2                  ; ADD STARTING T/G
005421  103202  177400                       BIC     #HIBYTE,R2              ; CLEAR UNUSED BITS
005423  103204  177400                       BIC     #HIBYTE,R4              ; CLEAR UNUSED BITS
005425  107042                               SUB     R4,R2                   ; FIND HOW MANY T/G'S TO SKIP
005426  025506                               CALL    COMPDP                  ; COMPUTE HOW MANY SECTORS
005427  100235                               MOV     R5,(R3)+                ; STORE IN LIST
```

```
                                        CLR     R5                      ; SET UP FOR MARKING EOL
                                        MOV     R5,(R3)+                ; STORE EOL
                                ;
                                ;       NOW SEE HOW MANY TIMES THE SETUP ROUTINE CAN GO THROUGH THE LIST
                                ;       WITHOUT EXCEEDING THE MAXIMUM SECTOR NUMBER
                                ;
                                        MOV     #S.BESS,R0              ; R0 WILL POINT TO THE MAX SEC NUMBER
                                        ADD     R1,R0                   ; R0 POINTS TO THE MAX SECTOR NUMBER
                                        MOV     S.TGOF-S.BESS(R0),R4    ; R4 HAS LO ORDER INITIAL OFFSET
                                        MOV     S.TGOF-S.BESS+1(R0),R5  ; R5 HAS HI ORDER INITIAL OFFSET
                                        PUSH    R1                      ; SAVE R1
                                                                                        MOV R1,-(SP)
                                        CLR     R2                      ; CLEAR LO ORDER MAX COUNT
                                        CLR     R3                      ; CLEAR HI ORDER MAX COUNT
                                        MOV     #S.TGSS-S.BESS,R1       ; R1 WILL POINT TO SECTOR OFFSET AREA
                                        ADD     R0,R1                   ; R1 POINTS TO SECTOR OFFSET AREA
                                13$:    MOV     R4,-(SP)                ; SAVE LO ORDER TOTAL ON STACK
                                        MOV     (R1)+,R4                ; GET LO ORDER SECTOR OFFSET
                                        BNE     14$                     ; IF NOT EOL, BRANCH
                                        MOV     R0,R1                   ; R1 WILL POINT TO START OF OFFSET LIST
                                        ADD     #S.TGSS-S.BESS,R1       ; R1 POINTS TO START OF OFFSET LIST
                                        MOV     (R1)+,R4                ; GET NEW OFFSET
                                14$:    ADD     (SP)+,R4                ; ADD TO LOW ORDER SECTOR TOTAL
                                        BCC     15$                     ; IF NO CARRY, BRANCH
                                        INC     R5                      ; PROPOGATE CARRY
                                15$:    CMP     1(R0),R5                ; SEE IF MAX EXCEEDED
                                        BCS     16$                     ; IF SO, BRANCH
                                                                                        BCC     .+2
                                                                                        BR      16$
                                        BNE     11$                     ; IF YOUR SURE IT'S OK, BRANCH
                                        CMP     (R0),R4                 ; SEE IF MAX EXCEEDED
                                        BCS     16$                     ; IF SO, BRANCH
                                                                                        BCC     .+2
                                                                                        BR      16$
                                11$:    ADD     #1,R2                   ; ADD 1 TO COUNT
                                        BCC     13$                     ; IF NO CARRY, BRANCH
                                        INC     R3                      ; PROPOGATE CARRY
                                        BR      13$                     ; BRANCH
                                16$:    POP     R1                      ; RESTORE R1
                                                                                        MOV (SP)+,R1
                                        MOV     R2,(R0)                 ; STORE LOW ORDER MAXIMUM COUNT
                                        MOV     R3,1(R0)                ; STORE HI MAXIMUM COUNT
                                        CLR     R2                      ; CLEAR R2
                                17$:    POP     R5                      ; RESTORE
                                                                                        MOV (SP)+,R5
                                        RETURN

                                TG$:    .BLKW   1
```

COMPDP - CALCULATE SECTORS/TRACKS OR SECTORS/GROUPS

```
                                        .SBTTL  COMPDP - CALCULATE SECTORS/TRACKS OR SECTORS/GROUPS
005506                                  COMPDP:
                                        ;
                                        ;       COMPUTE HOW MANY SECTORS ARE IN THE NUMBER OF TRACKS OR GROUPS
                                        ;       PASSED IN R2
                                        ;
005506  114005                                  CLR     R5                      ; CLEAR RUNNING TOTAL
005507  104114                                  MOV     (R1),R4                 ; GET SUBUNIT PARAMETERS
005510                                          ASSUME  S.PARM,0                ; ASSUME THAT S.PARM IS ZERO
005510  005521                                  BR      3$                      ; SEE IF IMMIDATE EXIT
005511  102204  000020                  1$:     BIT     #TRACKS,R4              ; SEE IF USING GROUPS OR TRACKS
005513  055517                                  BNE     2$                      ; IF TRACKS, BRANCH
005514  105305  004505                          ADD     SECGRP,R5               ; ADD SECTORS/GROUP TO RUNNING TOTAL
005516  005521                                  BR      3$                      ; BRANCH
005517  105305  004504                  2$:     ADD     SECTRK,R5               ; ADD SECTORS/TRACK TO RUNNING TOTAL
005521  117402                          3$:     DEC     R2                      ; DECREMENT COUNT
005522  035511                                  BPL     1$                      ; IF INCOMPLETE, BRANCH
005523  000000                                  RETURN                          ; RETURN TO CALLING PROGRAM
                                                .IF     LE,MAXADR-.
                                        MAXADR  =       .+1
                                                .ENDC
```

```
                                        .SBTTL  ********** OVERLAY MODULE GETSER - GET THE HDA AND DRIVE SERIAL NUMBER
                                                DMOVLY  GS,AREAI
                                                .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;*******************************************************************************
                                        ;
                                        ;
                                        ;
000035                                  GETSER  =       SCHAR1+1
                                        ;
                                        ;       GET HDA AND DRIVE SERIAL NUMBER (IF POSSIBLE) AND RETURN TO HOST
                                        ;
                                                .ENABL  LSB
004701                                          PUSH    R4                      ; SAVE SUBUNIT POINTER
004701  100464                                                                          MOV R4,-(SP)
004702  104650  000063  005342                 MOV     U.UNUM(R5),SUNIT        ; SAVE UNIT NUMBER
004705  104651  000063                         MOV     U.UNUM(R5),R1           ; R1 HAS UNIT NUMBER
004707  105201  000003                         ADD     #3,R1                   ; R1 HAS LAST UNIT NUMBER
004711  104207  000001                         MOV     #U.SUBP,R0              ; R0 WILL POINT TO SUBUNIT POINTERS
004713  105057                                 ADD     R5,R0                   ; R0 POINTS TO SUBUNIT POINTERS
004714  104274                          11$:   MOV     (R0)+,R4                ; R4 POINTS TO SUBUNIT
004715  075326                                 BMI     12$                     ; IF NO SUBUNIT, BRANCH
004716  104203  001750                         MOV     #1000.,R3               ; INITILIZE SEEK COUNTER
004720  100643  000001                         MOV     R3,S.SEEK(R4)           ; SAVE SEEK COUNTER
004722                                         PUSH    <R0,R1>                 ; SAVE POINTER AND COUNT
004722  100467                                                                          MOV R0,-(SP)
004723  100461                                                                          MOV R1,-(SP)
                                        .SBTTL  REPSER - FIND AND REPORT DRIVE AND HDA SERIAL NUMBER
                                        ;REPSER
                                        ;
                                        ;       FIND AND REPORT DRIVE AND HDA SERIAL NUMBER.  IF FRROR OCCURRS,
                                        ;       REPORT SERIAL NUMBER AS ZERO.  NO ERROR IS RETURNED FROM THIS ROUTINE
                                        ;
004724  021777                                 CALL    BULDUM                  ; BUILD DUMMY SDI BLOCK
004725  104641  000007                         MOV     S.SCHR(R4),R1           ; R1 POINTS TO SUBUNIT CHARACTERISTICS
004727  104610  000000  002173                 MOV     LBNCYL(R1),STACYL       ; MOVE LO STARTING CYL
004732  104610  000001  002174                 MOV     LBNCYL+1(R1),STACYL+1   ; MOVE HI STARTING CYL
004735  104617  000004                         MOV     RBNTRK(R1),R0           ; GET RBNS/TRACK
004737  103207  177600                         BIC     #HIBYTE!200,R0          ; CLEAR UNUSED BITS
004741  105617  000011                         ADD     LBNTRK(R1),R0           ; ADD LBNS/TRACK
004743  103207  177400                         BIC     #HIBYTE,R0              ; CLEAR UNUSED BITS
004745  104070  002177                         MOV     R0,LRDTRK               ; MOVE TO SECTORS PER TRACK
004747  114000  002200                         CLR     RBNFLG                  ; NO RBN FLAG
004751  104657  000057                         MOV     U.COPY(R5),R0           ; GET NUMBER OF XBN COPIES
004753  105077                                 ADD     R0,R0                   ; DOUBLE
004754  104070  005343                         MOV     R0,SERRTY               ; RETRY 2 TIMES THE NUMBER OF COPIES
004756  114007                          1$:    CLR     R0                      ; TO SET UP COPIES AND SETUP CALC
004757  100657  000060                         MOV     R0,U.CCOP(R5)           ; SAVE
004761  104641  000007                         MOV     S.SCHR(R4),R1           ; GET POINTER TO SUBUNIT CHARACTERISTICS
004763  114000  002175                         CLR     CURBN                   ; PUT LO ORDER XBN IN CURBN
004765  114000  002176                         CLR     CURBN+1                 ; PUT HI ORDER XBN IN CURBN+1
004767  104207  002173                  2$:    MOV     #CALCSC,R0              ; POINT TO CALCULATION AREA
004771  104641  000007                         MOV     S.SCHR(R4),R1           ; POINT TO SUBUNIT CHARACTERISTICS
004773  060020                                 XFC     CVT                     ; CONVERT
004774  023700                                 CALL    RECOVR                  ; CLEAR ALL POSSIBLE ERRORS
```

```
     71 004775  115002                          TST     R2                      ; SEE IF ERROR OCCURRED
     72 004776  055214                          BNE     8$                      ; IF SO, BRANCH
     73 004777  104203  001655                  MOV     #CR.INR,R3              ; SET UP FOR RECALIBRATE
     74 005001  021173                          CALL    TALK                    ; RECALIBRATE DRIVE
     75 005002  115002                          TST     R2                      ; SEE IF ERROR OCCURRED
     76 005003  055214                          BNE     8$                      ; IF SO, BRANCH
     77 005004  104300  002201  001673          MOV     CYL,LOCYL               ; MOVE LO CYL TO SEEK COMMAND
     78 005007  104300  002202  001674          MOV     CYL+1,LOCYL+1           ; MOVE TO SEEK COMMAND
     79 005012  104300  002203  001675          MOV     GROUP,LOCYL+2           ; MOVE GROUP TO SEEK COMMAND
     80 005015  104203  001636                  MOV     #CR.SEK,R3              ; POINT TO SEEK COMMAND
     81 005017  021173                          CALL    TALK                    ; SEND SEEK
     82 005020  115002                          TST     R2                      ; SEE IF ERROR OCCURRED
     83 005021  055214                          BNE     8$                      ; IF SO, BRANCH
     84 005022  104651  000011                  MOV     U.MSTO(R5),R1           ; GET MASTER SEEK TIMEOUT
     85 005024  100651  000005                  MOV     R1,U.TIMH(R5)           ; SAVE
     86 005026  114001                          CLR     R1                      ; FOR LOW ORDER TIMEOUT
     87 005027  100651  000006                  MOV     R1,U.TIML(R5)
     88 005031  104201  000310          3$:     MOV     #200.,R1                ; INNER LOOP TIMEOUT
     89 005033  117401                  4$:     DEC     R1                      ; DECREMENT INNER COUNTER
     90 005034  055033                          BNE     4$                      ; IF UNEXPIRED, BRANCH
     91 005035  020775                          CALL    RTDSL                   ; GET REAL TIME DRIVE STATE
     92 005036  115002                          TST     R2                      ; SEE IF ERROR OCCURRED
     93 005037  055214                          BNE     8$                      ; IF SO, BRANCH
     94 005040  115001                          TST     R1                      ; SEE IF READ/WRITE READY IS ASSERTED
     95 005041  075075                          BMI     5$                      ; IF SO, BRANCH
     96 005042                                  ASSUME  RWRDY,100000
     97 005042  114000  001105                  CLR     HRQ.02                  ; SETUP TO TALK TO HOST
     98 005044  114000  001106                  CLR     HRQ.03                  ; SETUP TO TALK TO HOST
     99 005046  114000  001107                  CLR     HRQ.04                  ; SETUP TO TALK TO HOST
    100 005050  104300  005342  001104          MOV     SUNIT,HRQ.01            ; MOVE UNIT NUMBER INTO SEND BUFFER
    101 005053  104207  060007                  MOV     #T4SOFT,R0              ; MOVE REQUEST NUMBER TO R0
    102 005055  021053                          CALL    HOSTRQ                  ; TELL THE HOST I'M ALIVE
    103 005056  104651  000006                  MOV     U.TIML(R5),R1           ; GET TIMEOUT
    104 005060  107201  000001                  SUB     #1,R1                   ; DECREMENT TIMEOUT
    105 005062  100651  000006                  MOV     R1,U.TIML(R5)           ; SAVE
    106 005064  045031                          BCC     3$                      ; LOOP IF INCOMPLETE
    107 005065  104651  000005                  MOV     U.TIMH(R5),R1           ; GET HI ORDER TIMEOUT
    108 005067  107201  000001                  SUB     #1,R1                   ; DECREMENT TIMEOUT
    109 005071  100651  000005                  MOV     R1,U.TIMH(R5)           ; SAVE
    110 005073  045031                          BCC     3$                      ; IF TIMOUT UNEXPIRED, BRANCH
    111 005074  005214                          BR      8$                      ; ERROR
    115 005075  104200  100000  005344  5$:     MOV     #RSTOP,SERCHN+RW.STAT   ; MOVE LAST CHAIN FLAG TO CHAIN
    124 005100  104300  002175  005346          MOV     CURBN,SERCHN+RW.LOW     ; MOVE TO OUTPUT
    125 005103  104641  000007                  MOV     S.SCHR(R4),R1           ; R1 POINTS TO SUBUNIT CHARACTERISTICS
    126 005105  104611  000002                  MOV     HIXBN(R1),R1            ; GET HI XBN BITS
    127 005107  110601                          ROR     R1                      ; ROTATE TO CORRECT POSITION
    128 005110  110601                          ROR     R1                      ; ROTATE TO CORRECT POSITION
    129 005111  110601                          ROR     R1                      ; ROTATE TO CORRECT POSITION
    130 005112  110601                          ROR     R1                      ; ROTATE TO CORRECT POSITION
    131 005113  103201  170377                  BIC     #HBLONB,R1              ; CLEAR UNUSED BITS
    132 005115  101201  120000                  BIS     #HD.XBN,R1              ; MAKE A XBN
    133 005117  105301  002176                  ADD     CURBN+1,R1              ; SET IN HI XBN BITS
    134 005121  104010  005347                  MOV     R1,SERCHN+RW.HI         ; SAVE
    135 005123  104300  002204  005350          MOV     TRACK,SERCHN+RW.CMD     ; MOVE TRACK TO CHAIN
    136 005126  101200  013400  005350          BIS     #RREAL,SERCHN+RW.CMD    ; SET IN REAL TIME COMMAND
    137 005131  104200  002207  005351          MOV     #DUMSDI,SERCHN+RW.SDI   ; POINT TO DUMMY SDI BLOCK
    138 005134  104652  000025                  MOV     U.MASK(R5),R2           ; R2 HAS UDA PORT MASK
```

```
                                                MOV     #SERCHN,R0              ; R0 POINTS TO CHAIN
                                                XFC     WAITSI                  ; WAIT FOR SECTOR OR INDEX
                                                TST     R1                      ; SEE IF ERROR OCCURRED
                                                BNE     8$                      ; IF SO, BRANCH
                                                XFC     XREAD                   ; READ THE SECTOR
                                                CMP     #4,R1                   ; SEE IF XBN HEADER COMPARE FAILURE
                                                BEQ     7$                      ; IF SO, TRY NEXT COPY
                                                TST     R1                      ; SEE IF AN ERROR OCCURRED
                                                BNE     8$                      ; IF SO, BRANCH
                                                BIT     #ECCFLG,SERCHN+RW.STAT  ; SEE IF BUFFER HAS AN ECC ERROR
                                                BEQ     6$                      ; IF NOT, BRANCH
                                                MOV     #SERCHN,R0              ; POINT TO CHAIN
                                                XFC     ECC                     ; CORRECT THE BUFFER
                                                TST     R1                      ; SEE IF ERROR
                                                BNE     7$                      ; IF SO, BRANCH
                                                CMP     U.ECCT(R5),R0           ; SEE IF CORRECTIONS EXCEED THRESHOLD
                                        ;
                                        ;       NOTE: I ALLOW 1 ERROR MORE THAN SPEC'D ECC THRESHOLD
                                        ;
                                                BMI     7$                      ; IF SO, BRANCH
                                        6$:     MOV     #SERSEC,R0              ; R0 POINTS TO BUFFER
                                                CALL    CMPEDC                  ; COMPUTE EDC OVER BUFFER
                                                CMP     R2,SERSEC+BF.EDC        ; SEE IF EDC'S MATCH
                                                BEQ     9$                      ; IF SO, YOU'VE GO IT!!!!!!!!!!!!!
                                        7$:     MOV     U.CCOP(R5),R1           ; GET CURRENT COPY NUMBER
                                                INC     R1                      ; INCREMENT COUNT
                                                MOV     R1,U.CCOP(R5)           ; SAVE
                                                CMP     U.COPY(R5),R1           ; SEE IF ALL COPIES TRIED
                                                BEQ     8$                      ; IF SO, BRANCH
                                                MOV     S.SCHR(R4),R0           ; R1 POINTS TO SUBUNIT CHARACTERISTICS
                                                ADD     FCTSIZ(R0),CURBN        ; ADD TO CURRENT BN
                                                BCC     2$                      ; CALCULATE AND TRY NEXT SECTOR
                                                INC     CURBN+1                 ; PROPOGATE CARRY
                                                BR      2$                      ; BRANCH
                                        8$:     DEC     SERRTY                  ; DECREMENT RETRY COUNT
                                                BNE     1$                      ; IF UNEXHAUSTED, BRANCH
                                        ;
                                        ;       COULDN'T GET THE SERIAL NUMBER -- REPORT 0'S AND 512 DRIVE
                                        ;
                                                CLR     SERSEC+2                ; ZERO HDA SER #
                                                CLR     SERSEC+3                ; ZERO HDA SER #
                                                CLR     SERSEC+4                ; ZERO HDA SER #
                                                CLR     SERSEC+5                ; ZERO HDA SER #
                                                BR      10$                     ; ASSUME 512 DRIVE
                                        9$:     CMP     #MOD512,SERSEC          ; SEE IF 512 BYTE MODE DRIVE
                                                BEQ     10$                     ; IF SO, BRANCH
                                                MOV     (R4),R0                 ; GET UNIT PARAMETERS
                                                ASSUME  S.PARM,0
                                                BIT     #DCYLS,R0               ; SEE IF USING THE DIAGNOSTIC AREA
                                                BNE     10$                     ; IF SO, OK TO TEST, BRANCH
                                                DEVFTL  21,SERSEC               ; SET UP REPORT
                                                        MOV     #ER21,HRQ.04
                                                        MOV     SERSEC,HRQ.05
                                                        MOV     #21!FTLDEV+4000.,R2
                                                        MOV     R2,HRQ.02
                                                        MOV     #.,HRQ.01
                                                        MOV     #ERRMC,HRQ.RQ
```

```
    208 005260  104300  005342  001106          MOV     SUNIT,HRQ.03        ; MOVE UNIT NUMBER TO MESSAGE
    209 005263  104307  001103                  MOV     HRQ.RQ,R0           ; GET REQUEST NUMBER
    210 005265  021053                          CALL    HOSTRQ              ; REPORT TO HOST
    211 005266  101200  000002  002223          BIS     #DIE,M.PARM         ; DO NOT TEST
    212 005271  104300  005342  001104  10$:     MOV     SUNIT,HRQ.01        ; MOVE IN UNIT NUMBER
    213 005274  104300  004516  001105          MOV     DSERNM,HRQ.02       ; MOVE DRIVE SERIAL NUMBER TO BUFFER
    214 005277  104300  004517  001106          MOV     DSERNM+1,HRQ.03     ; MOVE DRIVE SERIAL NUMBER TO BUFFER
    215 005302  104300  004520  001107          MOV     DSERNM+2,HRQ.04     ; MOVE DRIVE SERIAL NUMBER TO BUFFER
    216 005305  104300  005354  001110          MOV     SERSEC+2,HRQ.05     ; MOVE HDA SERIAL NUMBER TO BUFFER
    217 005310  104300  005355  001111          MOV     SERSEC+3,HRQ.06     ; MOVE HDA SERIAL NUMBER TO BUFFER
    218 005313  104300  005356  001112          MOV     SERSEC+4,HRQ.07     ; MOVE HDA SERIAL NUMBER TO BUFFER
    219 005316  104300  005357  001113          MOV     SERSEC+5,HRQ.08     ; MOVE HDA SERIAL NUMBER TO BUFFER
    220 005321  104207  060004                  MOV     #T4UPRM,R0          ; MOVE IN REQUEST
    221 005323  021053                          CALL    HOSTRQ              ; REPORT
    222 005324                                  POP     <R1,R0>             ; RESTORE POINTER AND COUNT
        005324  104261                                                                  MOV (SP)+,R1
        005325  104267                                                                  MOV (SP)+,R0
    223 005326  115400  005342          12$:    INC     SUNIT               ; INCREMENT ACTIVE SUBUNIT
    224 005330  106010  005342                  CMP     R1,SUNIT            ; SEE IF ALL SUBUNITS REPORTED
    225 005332  044714                          BCC     11$                 ; IF NOT, BRANCH
    226 005333                                  POP     R4                  ; RESTORE SUBUNIT POINTER
        005333  104264                                                                  MOV (SP)+,R4
    227 005334  114002                          CLR     R2                  ; NO ERRORS
    228 005335  114001                          CLR     R1                  ; IMMIDATE CALL
    229 005336  104207  000036                  MOV     #INITD,R0           ; INITD NEXT ROUTINE
    230 005340  024172                          CALL    DSABLE              ; DISABLE ERROR RECOVERY
    231 005341  003231                          BR      JMPRET              ; RETURN TO SEQUENCER
    232                                         .DSABL  LSB
```

ASSOCIATED VARIABLES FOR GETTING THE HDA SERIAL NUMBER

```
      1                                 .SBTTL  ASSOCIATED VARIABLES FOR GETTING THE HDA SERIAL NUMBER
      2 005342  000000                  SUNIT:  .WORD   0                       ; THE ACTIVE UNIT NUMBER
      3 005343  000000                  SERRTY: .WORD   0                       ; RETRY COUNT
      4 005344                          SERCHN:
      5                                         ; THE CHAIN FOR READING SECTOR 0 (RCT)
     13 005344  000000                          .WORD   0
     14 005345  005352                          .WORD   SERSEC                  ; POINTER TO THE SECTOR
     15 005346  000000                          .WORD   0
     16 005347  000000                          .WORD   0
     17 005350  000000                          .WORD   0
     18 005351  000000                          .WORD   0
     22 005352                          SERSEC:                                ; 269 WORD BUFFER FOR READ
     23                                         .IF     LE,MAXADR-<.+269.>
     24                                 MAXADR  =       .+269.
     25                                         .ENDC
```

```
                                .SBTTL  ********** OVERLAY MODULE INITD - INITILIZE THE DRIVE  PARAMETERS FOR TESTING
005352                                  DMOVLY  ID,AREAI
005352  000105                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;
                                ;
                                ;
000036                          INITD   =       GETSER+1
                                ;
                                ;       BRING DRIVE ONLINE, CLEAR ALL ERRORS (IF ANY) AND CHANGE THE
                                ;       MODE TO THE REQUIRED VALUES
                                ;
                                        .ENABL  LSB
004701  104207  000004                  MOV     #4,R0           ; MAXIMUM OF 4 SUBUNITS
004703  104201  000001                  MOV     #U.SUBP,R1      ; R1 WILL POINT TO SUBUNIT POINTERS
004705  105051                          ADD     R5,R1           ; R1 POINTS TO SUBUNIT POINTERS
004706  104212                  1$:     MOV     (R1)+,R2        ; GET POINTER TO SUBUNIT INFO
004707  074725                          BMI     2$              ; IF UNIT NOT TESTED, BRANCH
004710  104123                          MOV     (R2),R3         ; GET SUBUNIT PARAMETERS
004711                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
004711  102203  000100                  BIT     #SEQSEK,R3      ; SEE IF SEQUENTIAL SEEKS
004713  054717                          BNE     7$              ; IF SO, BRANCH
004714  114000  002217                  CLR     SCR1            ; FLAG AS RANDOM LBNS
004716  004722                          BR      8$              ; BRANCH
004717  104200  177777  002217  7$:     MOV     #-1,SCR1        ; FLAG AS SEQUENTIAL LBNS
004722  102203  040000          8$:     BIT     #INITW,R3       ; SEE IF SUBUNIT WILL BE INITALLY WRITTEN
004724  054730                          BNE     3$              ; IF SO, BRANCH
004725  117407                  2$:     DEC     R0              ; DECREMENT COUNT
004726  054706                          BNE     1$              ; IF NOT ALL SUBUNITS CHECKED, BRANCH
004727  004741                          BR      4$              ; BRANCH
004730  104657  000046          3$:     MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
004732  101207  040000                  BIS     #INITW,R0       ; FLAG UNIT AS SUBUNITS GET INITIALLY WRITTEN
004734  100657  000046                  MOV     R0,U.PARM(R5)   ; SAVE
004736  101200  100000  002223          BIS     #IWIPRG,M.PARM  ; FLAG MASTER PARAMETERS AS INITIAL WRITE TO BE DONE
004741  104207  000004          4$:     MOV     #4,R0           ; MAXIMUM OF FOUR SUBUNITS
004743  104201  000005                  MOV     #U.SUBP+4,R1    ; R1 WILL POINT TO AFTERS SUBUNIT POINTERS
004745  105051                          ADD     R5,R1           ; R1 POINTS TO AFTERS SUBUNIT POINTERS
004746  114003                          CLR     R3              ; INITILIZE SUBUNIT PROTECTION FLAGS
004747  110203                  5$:     ROL     R3              ; ROTATE R3
004750  103203  000001                  BIC     #1,R3           ; CLEAR LO BIT
004752  104412                          MOV     -(R1),R2        ; GET SUBUNIT POINTER
004753  075022                          BMI     6$              ; IF NO SUBUNIT, BRANCH
004754  104122                          MOV     (R2),R2         ; GET SUBUNIT STATUS
004755                                  ASSUME  S.PARM,0        ; ASSUME THAT S.PARM IS ZERO
004755  102202  000100                  BIT     #SEQSEK,R2      ; SEE IF SEQUENTIAL SEEKS
004757  054764                          BNE     9$              ; IF SO, BRANCH
004760  115000  002217                  TST     SCR1            ; SEE IF ALL SUBUNITS RANDOM SEEKS
004762  054767                          BNE     10$             ; IF NOT, BRANCH
004763  005015                          BR      11$             ; BRANCH
004764  115000  002217          9$:     TST     SCR1            ; SEE IF ALL SUBUNITS SEQUENTIAL SEEKS
004766  055015                          BNE     11$             ; IF SO, BRANCH
004767                          10$:    DEVFTL  56,#SER20       ; REPORT ERROR
004767  104200  003604  001107                                          MOV     #ER56,HRQ.04
004772  104200  004047  001110                                          MOV     #SER20,HRQ.05
```

```
        004775  104202  047730                                          MOV     #56!FTLDEV+4000.,R2
        004777  104020  001105                                          MOV     R2,HRQ.02
        005001  104200  005001  001104                                  MOV     #.,HRQ.01
        005004  104200  060014  001103                                  MOV     #ERRMC,HRQ.RQ
    72  005007                          ENDERR  0
        005007  114000  002230                                          CLR     ERRPOS          ; CLEAR THE POSITION
    73  005011  101200  000002  002223          BIS     #DIE,M.PARM     ; FLAG INITIALIZATION ERROR
    74  005C14  005040                  BR      DRVEXT          ; EXIT
    75  005015  102202  004000          11$:    BIT     #RONLY,R2       ; SEE IF READ ONLY DRIVE
    76  005017  015022                  BEQ     6$              ; IF NOT READ ONLY, BRANCH
    77  005020  101203  000001          BIS     #1,R3           ; SET READ ONLY BIT
    78  005022  117407                  6$:     DEC     R0              ; DECREMENT COUNT
    79  005023  054747                  BNE     5$              ; IF COUNT UNEXPIRED, BRANCH
    80  005024  110203                  ROL     R3              ; ROTATE MASK TO CORRECT POSITION
    81  005025  110203                  ROL     R3              ; ROTATE MASK TO CORRECT POSITION
    82  005026  110203                  ROL     R3              ; ROTATE MASK TO CORRECT POSITION
    83  005027  110203                  ROL     R3              ; ROTATE MASK TO CORRECT POSITION
    84  005030  103203  177417          BIC     #LBHINB,R3      ; CLEAR UNUSED BITS
    85  005032  100653  000045          MOV     R3,U.WPRT(R5)   ; SAVE
    86  005034  104207  000001          MOV     #SETUP,R0       ; SETUP NEXT ROUTINE CALLED
    87  005036  104051                  MOV     R5,R1           ; MAKE R1 NON-ZERO (DEFERRED CALL)
    88  005037  114002                  CLR     R2              ; NO ERRORS
    89  005040  003231          DRVEXT: BR      JMPRET          ; RETURN TO SEQUENCER
    90                                  .DSABL  LSB
    94                                  .IF     LE,MAXADR-.
    95                          MAXADR  =       .+1
    96                                  .ENDC
   113  005041                          DMEND
        005041  000105                  .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
   114          000001                  .END
```

SYMBOL TABLE

```
AFTOP   004632   COPLP0  004550   D$      005521   ER44    003127   GETSUB= 000210
AFTWRT= 000011   COPLP1  004555   D.LIMT= 000001   ER45    003162   GETU    004755
ALLOCM  004674   COPLP2  004565   D.SCHR= 000002   ER47    003214   GMPARM  004701
AREAI = 004701   COUNTD  005055   ECC   = 000015   ER48    003243   GOINIT  004223
AREA0 = 004413   CPYBEX  005775   ECCCHK= 010000   ER49    003270   GONE    003202
AREA1 = 004602   CR.CLR  001624   ECCFLG= 010000   ER5     000365   GORCLB  004215
AREA2 = 005043   CR.DIS  001650   ECCRSH= 000002   ER50    003315   GORTRY  004201
ATTN  = 000002   CR.ERR  001643   ECHO  = 000010   ER51    003362   GOSEEK  004207
AVAIL = 000100   CR.GCR  004453   ECHOC = 000350   ER52    003405   GOTOBE  005074
BBLOP   005741   CR.GST  001617   EDCLOP  001155   ER53    003465   GOTOFS  005077
BEEXT   005251   CR.INR  001655   ENABLE  004163   ER54    003502   GO4IT   003214
BESDWN  005134   CR.MOD  001631   EOC   = 100000   ER55    003552   GROUP   002203
BESUP   005250   CR.ONL  001612   ERCOV = 000016   ER56    003604   GRPCYL= 000002
BEUSED= 000040   CR.RUN  004465   ERECOV= 000006   ER57    003644   GRPOFF= 000011
BF.DAT= 000000   CR.SCR  004440   EREXT   004504   ER58    003717   GRP$    005513
BF.ECC= 000401   CR.SEK  001636   ERHARD= 100000   ER59    003750   GST     001670
BF.EDC= 000400   CSCEXT  004510   ERLEV = 000002   ER6     000562   HBHINB= 007777
BLDBB   005546   CURBN   002175   ERMASK= 100000   ER62    003765   HBLONB= 170377
BLDUNT  005365   CVT   = 000020   ERMODE  002226   ER63    004115   HDRPRE= 000005
BLKCHK  001725   CYL     002201   ERR     001676   ER64    004145   HD.BAD= 110000
BREAK = 000000   CYLBN   005337   ERRLEV  001677   ER68    004175   HD.DBN= 140000
BSUSEX  006006   CYLSCR  005575   ERRMC = 060014   ER69    004220   HD.LBN= 000000
BUFARA= 005230   C2DFTL= 100000   ERRMES= 060013   ER7     000615   HD.PRV= 050000
BUFFLG= 040000   C2HARD= 040000   ERRPOS  002230   ER70    004261   HD.RBN= 060000
BUFSIZ= 000043   DATCMP= 000002   ERSOFT= 140000   ER71    004275   HD.REV= 030000
BUILDP= 000007   DATPRE= 000005   ER1     000000   ER72    004312   HD.XBN= 120000
BULDUM  001777   DBNCYL= 000022   ER10    000770   ER73    004342   HIBN    002172
CALC    002027   DCEXT   004743   ER1000  005066   ER74    004366   HIBYTE= 177400
CALCSC  002173   DCLOCK= 000004   ER11    001044   ER75    004413   HICYL = 000001
CBB2    004623   DCLR    001667   ER12    001113   ER76    004444   HIDBN = 000003
CHAINS  002232   DCMPAL= 000001   ER13    001406   ER77    004523   HILBN = 000002
CHGMOD= 000201   DCMPYS  004737   ER14    001435   ER78    004557   HIMEM = 007774
CHKBB   005052   DCREAD  004731   ER15    001503   ER8     000631   HIRBN = 000003
CHKBES  005411   DCYLS = 020000   ER16    001553   ER9     000745   HISEED  002225
CHKCYL  005234   DEBUG = 000000   ER17    001604   EXIT  = 000021   HIXBN = 000002
CHKDN   004064   DIE   = 000002   ER17A   001634   FB.DAT= 000000   HOSTRQ  001053
CHKECC= 000015   DINIT = 000011   ER18    001702   FB.EDC= 000400   HRDREV= 000003
CHKEDC= 000014   DIREC = 010000   ER19    001754   FCTSIZ= 000010   HRQ.RQ  001103
CHKTG   005223   DIS     001662   ER2     000117   FIRSTU= 007702   HRQ.01  001104
CHKUP   004114   DISCON= 000204   ER20    002056   FIXPAT  004673   HRQ.02  001105
CHNMAX  002234   DONE  = 060016   ER21    002106   FNDBES  005266   HRQ.03  001106
CHRRES= 000170   DOWNG   005342   ER23    002157   FNDLOP  005276   HRQ.04  001107
CLCLBN  005144   DOWNT   005200   ER24    002272   FNDRER  004641   HRQ.05  001110
CLCMAX  005737   DRC     001666   ER25    002337   FORMAT= 000001   HRQ.06  001111
CMPDAT= 000020   DRINIT= 040000   ER26    002366   FSTOP = 100000   HRQ.07  001112
CMPEDC  001146   DROP  = 100000   ER28    002415   FTIME = 001000   HRQ.08  001113
CMPERR  004515   DRPALL= 000026   ER3     000166   FTLDEV= 040000   HRQ.09  001114
CMP2    001765   DRTYPE= 000007   ER33    002452   FTLSYS= 000000   HRQ.10  001115
CMXEX   004707   DRVCLR= 000005   ER34    002540   FT.BUF= 000000   HRQ.11  001116
CNTLOP  005047   DRVEXT  005040   ER35    002563   FT.HI = 000001   HRQ.12  001117
COMCHR= 000030   DRVID = 000004   ER36    002612   FT.LOW= 000002   HRQ.13  001120
COMPAR= 000006   DRVONL= 000213   ER37    002634   FWRD    004514   HRQ.14  001121
COMPDP  005506   DRVRUN= 000014   ER38    002660   GCR     004473   HRQ.15  001122
COMPLT= 000176   DSABLE  004172   ER4     000271   GETCHR= 000207   HRQ.16  001123
COMPSC  005710   DSERNM  004516   ER40    002705   GETMEM  004616   HRQ.17  001124
COPBB   005721   DUMSDI  002207   ER41    002775   GETSER= 000035   HRQ.18  001125
COPFIN  004570   DUPPKT= 060000   ER42    003066   GETSTA= 000011   HRQ.19  001126
```

SYMBOL TABLE

```
HRQ.20  001127      L2.SLN= 000001      ONL     001700      OVL.SB= 000537      PAT12   002454
HRQ.21  001130      MASK    005025      ONLYCL= 000200      OVL.SC= 000615      PAT13   002476
HRQ.22  001131      MAXADR= 006024      OTABLE  004257      OVL.SK= 000323      PAT14   002520
HRQ.23  001132      MAXDBN  004502      OUTO    005304      OVL.SO= 000222      PAT15   002525
HRQ.24  001133      MAXMSK= 177774      OVE.AW= 004413      OVL.SP= 000027      PAT2    002303
HRQ.25  001134      MAXMUM  005305      OVE.BP= 004413      OVL.ST= 000471      PAT3    002306
HRQ.26  001135      MAXSEC  002235      OVE.CC= 004701      OVL.SU= 000167      PAT4    002311
HRQ.27  001136      MAXSND= 001750      OVE.CD= 004413      OVL.SO= 001101      PAT5    002333
HRQ.28  001137      MAXSUB= 000010      OVE.CK= 004413      OVL.S1= 000624      PAT6    002355
HRQ.29  001140      MEDTYP= 000006      OVE.DA= 004413      OVL.TS= 000371      PAT7    002377
HRQ.30  001141      MEMPOL  002227      OVE.EC= 004413      OVL.W = 000404      PAT8    002402
HRQ.31  001142      MESSAG= 060015      OVE.ED= 004413      OVL...= 026746      PAT9    002424
HRQ.32  001143      MICREV= 000003      OVE.GS= 004701      OVSTRT= 007774      PHYSA = 001000
HRQ.33  001144      MINADR= 006232      OVE.ID= 004701      OVS.AW= 035400      PLOAD   003211
HRQ.34  001145      MKLOOP  004642      OVE.IN= 004701      OVS.BP= 033606      PNUM    002221
ID      005556      MOD     001664      OVE.LM= 004413      OVS.CC= 050424      PREVBE  005325
INCODT= 000000      MOD512= 126736      OVE.MN= 000714      OVS.CD= 043764      PTABLE  004245
INDEX   002206      MOD576= 074161      OVE.MS= 000000      OVS.CK= 042174      QDA   = 000000
INITD = 000036      MORE    004713      OVE.NL= 004413      OVS.DA= 050232      RANDOM  004764
INITW = 040000      MOVDBN  002123      OVE.NO= 004602      OVS.EC= 043232      RBLOCK  004537
INR     001671      MOVOUT  002164      OVE.R = 004413      OVS.ED= 041216      RBNBN = 000200
INS     001672      MOV2    005420      OVE.RB= 005043      OVS.GS= 055326      RBNFLG  002200
INSEEK= 000012      MRD   = 000016      OVE.RC= 005230      OVS.ID= 056452      RBNTRK= 000004
INSET = 000027      MSSG$ = 000000      OVE.RT= 005043      OVS.IN= 050360      RBNTXT  005472
INTEDC= 000105      MS1     005527      OVE.RV= 004413      OVS.LM= 044566      RBUFLN= 000415
INTINP= 000001      MS2     005545      OVE.SB= 005043      OVS.MN= 001040      RCBREQ= 002000
IRECLB= 000216      MS3     005616      OVE.SC= 004413      OVS.MS= 013260      RCONT = 000000
ISUEXT  005551      MS4     005701      OVE.SK= 005230      OVS.NL= 043422      RCTCPS= 000001
ISUSEK  005473      MWR   = 000017      OVE.SO= 004701      OVS.NO= 027534      RCTCSZ= 000014
IWIPRG= 100000      M.PARM  002223      OVE.SP= 004701      OVS.R = 037010      RCTL$   005523
JMPRET  003231      NEWDNT  005424      OVE.ST= 005043      OVS.RB= 030236      RCV   = 000005
LASTU   004477      NEWEXT  004571      OVE.SU= 004413      OVS.RC= 050072      RCVERR= 000400
LBHINB= 177417      NEWLEV= 000017      OVE.SO= 004701      OVS.RT= 032074      RCVRDY= 000001
LBLONB= 177760      NEWOP = 000002      OVE.S1= 004701      OVS.RV= 044722      RDSTAT  001007
LBNCYL= 000000      NEWSUB= 004000      OVE.TS= 005230      OVS.SB= 030576      READ  = 000012
LBNHST= 000012      NEWUPT  005376      OVE.W = 004413      OVS.SC= 037564      RECALB= 000025
LBNTRK= 000011      NEXTBE  005361      OVL.AW= 000604      OVS.SK= 046242      RECOVR  003700
LEVNZR  004531      NLBEXT  005135      OVL.BP= 000271      OVS.SO= 051010      REDWRT= 000100
LEVUSD= 020000      NLEV    004473      OVL.CC= 000143      OVS.SP= 050732      RETRY = 001000
LINKLN= 000007      NOBB    005773      OVL.CD= 000301      OVS.ST= 032424      RETS  = 000001
LNCONT  002107      NOBORO  005111      OVL.CK= 000417      OVS.SU= 027156      REVCT = 000022
LNCYL   002101      NOEXTR  005756      OVL.DA= 000053      OVS.SO= 051454      REVEC = 000400
LOADED  003175      NOSEEK  005417      OVL.EC= 000074      OVS.S1= 053656      REVS  = 000007
LOBYTE= 000377      NOSUB   003675      OVL.ED= 000367      OVS.TS= 047110      REVSOK  004525
LOCYL   001673      NOSUN   006004      OVL.GS= 000452      OVS.W = 034370      RM    = 000004
LONG  = 001654      NOU     005456      OVL.ID= 000141      OV... = 027366      RNDBE = 000003
LONGTO= 000001      NOUNIT  002560      OVL.IN= 000022      PARTI   004255      RNDTG = 000005
LOSEED  002224      NUMBB   004501      OVL.LM= 000056      PART0   004245      RNDO    004667
LRDTRK  002177      NUML2S= 000010      OVL.MN= 005110      PART1   004247      RONLY = 004000
LSTMOD= 000021      NUMOVL= 000037      OVL.MS= 005737      PART2   004251      ROOT    002532
LSTOVL  002222      NUMPTR= 000006      OVL.NL= 000161      PART3   004253      ROOTLP  002535
LSTTRK  005511      NXTLEV= 010000      OVL.NO= 000241      PATEXT  004675      RREAL = 013400
L$      005517      NXTRCT  004774      OVL.R = 000266      PATPTR  002236      RSTOP = 100000
L2.EOF= 000004      NXTTRK  005433      OVL.RB= 000160      PAT0    002256      RTDS    000746
L2.ERS= 000003      OCNT$ = 000037      OVL.RC= 000060      PAT1    002300      RTDSL   000775
L2.OPC= 000000      ONEPAT  004564      OVL.RT= 000154      PAT10   002427      RTRIES= 001000
L2.RLN= 000002      ONEPAX  004501      OVL.RV= 000550      PAT11   002451      RUN     004472
```

SYMBOL TABLE

```
RVFAIL  005125        SER15   005311        STACYL  002173        TG$     005505        U.MASK= 000025
RWRDY = 100000        SER16   005333        START   004521        THREBE  005071        U.MBN = 000051
RW.ANG= 000006        SER17   005356        STATUS= 000007        TLEN.U= 000072        U.MLEV= 000031
RW.BUF= 000001        SER18   005410        STOP    002573        TLKHST  001031        U.MSEC= 000022
RW.CMD= 000004        SER18A  005432        STSERR  005417        TMEMPL  004701        U.MSTO= 000011
RW.HI = 000003        SER18B  005426        STSEXT  005617        TO      005031        U.NEXT= 000000
RW.LOW= 000002        SER18C  005422        STSRES= 000366        TRACK   002204        U.NFUN= 000013
RW.SDI= 000005        SER18D  005416        ST.C  = 000002        TRACKS= 000020        U.NSEC= 000021
RW.STA= 000000        SER18E  005341        ST.CON= 000002        TRAV    005156        U.PARM= 000046
SBCRES= 000167        SER18F  004461        ST.DB = 001000        TRKGRP= 000003        U.PAT = 000014
SCHAR0= 000033        SER19   000470        ST.DE = 000200        TRK$    005507        U.PCTG= 000017
SCHAR1= 000034        SER2    004665        ST.DF = 000020        TROOT   004640        U.RBN = 000055
SCHRCP  005210        SER20   004047        ST.DR = 000040        TRYAGN  005057        U.RCOV= 000047
SCR     004474        SER21   000661        ST.EL = 000010        TSTNEC  005424        U.RRTY= 000010
SCR1    002217        SER22   000051        ST.ERR= 000002        T1MSIZ= 060000        U.RTRY= 000030
SCR2    002220        SER24   001326        ST.FO = 002000        T2CMD = 060002        U.RVER= 000062
SCTWRD= 000377        SER25   001354        ST.MOD= 000001        T2DLL = 060001        U.RWER= 000061
SDIVER= 000000        SER26   003035        ST.MSK= 000000        T4BB1 = 060005        U.RWTO= 000012
SDI$  = 000003        SER3    004707        ST.OA = 000200        T4BB2 = 060006        U.SDIL= 000034
SEARCH  004635        SER32   003051        ST.PE = 000040        T4MPRM= 060003        U.SDIS= 000033
SECCHK= 000013        SER36   004242        ST.PS = 000002        T4MXFR= 060011        U.SDI2= 000035
SECCYL  004506        SER39   002477        ST.RE = 000100        T4SEEK= 06001C        U.SRTY= 000007
SECGRP  004505        SER4    004726        ST.REQ= 000001        T4SOFT= 060007        U.SUBP= 000001
SECMAX  002233        SER40   005435        ST.RR = 000100        T4UPRM= 060004        U.SUBU= 000050
SECPTR  004702        SER41   005457        ST.RTY= 000003        UCURSR  004476        U.TIMH= 000005
SECTOR  002205        SER42   002200        ST.RU = 000001        UDADM4= 001000 G      U.TIML= 000006
SECTRK  004504        SER43   002214        ST.SR = 000020        UDA50 = 000001        U.TSEC= 000023
SEC512= 000400        SER44   002231        ST.S7 = 000400        UDA52 = 000000        U.UNUM= 000063
SEC576= 000440        SER45   002243        ST.UNT= 000000        ULOP    005373        U.WPRT= 000045
SEEK  = 000023        SER5    004747        ST.WE = 000010        UMASK   004500        U.WRIT= 000026
SEKCNT  005275        SER6    004770        SUBTRV  005167        UNADDR= 170000        VALBIT= 077177
SEKEXT  005423        SER7    005010        SUBUNT  004475        UNITS   005345        VALID = 100000
SEKINP= 002000        SER8    005027        SUNIT   005342        UNSSUC= 000175        WAITSI= 000012
SEKOUT  005422        SER9    005050        SUSETL  005476        UP      005154        WBLOCK  004637
SEKREG= 004000        SETSEC  005500        SWAPBE  004775        UPG     005262        WBUFLN= 000401
SEKTST= 000024        SETUP = 000001        S.BADP= 000012        UPT     005121        WCHECK= 000010
SEND  = 000004        SHRTTO= 000000        S.BESS= 000013        UREAD = 000013        WCHKAL= 000004
SEQBE = 000004        SLOP    005507        S.LETR= 000005        UTOTST= 060012        WCONT = 040000
SEQLP   002601        SMASKL  004712        S.MCNT= 000013        UWRITE= 000014        WONLY = 002000
SEQNCR  002577        SMASKX  004716        S.MEGR= 000010        U.CBN = 000053        WREAL = 122400
SEQSEK= 000100        SNDAGN  001060        S.MEGW= 000011        U.CCNT= 000015        WRITBT  001665
SEQTG = 000006        SNDEX   004572        S.PARM= 000000        U.CCOP= 000060        WRITE = 000010
SERCHN  005344        SNDONE= 001616        S.PAT = 000004        U.CCYL= 000064        WSTOP = 140000
SERRTY  005343        SORT  = 000032        S.SCHR= 000007        U.CGRP= 000066        XBNCYL= 000021
SERSEC  005352        SORTBB  005025        S.SDCL= 000002        U.COPY= 000057        XFERRT= 000000
SER0    004630        SORTBE  004735        S.SEEK= 000001        U.CSEC= 000024        XOPLP0  004462
SER1    004646        SORTTG  005074        S.TGOF= 000015        U.CTRK= 000020        XOPLP1  004467
SER10   005205        SPINUP= 000031        S.TGSS= 000017        U.ECCT= 000032        XOPLP2  004502
SER11   005220        SS    = 000001        S.TRKL= 000006        U.ELEV= 000027        XREAD = 000002
SER12   005236        ST      001702        TALK    001173        U.LCYL= 000067        XWRITE= 000003
SER13   005250        STACK   000745        TGS     004507        U.LGRP= 000071        ZEREDC  001165
SER14   005263

. ABS.  056754      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  25264 WORDS  ( 99 PAGES)
```

UDAT4 DISK EXERCISER DMACR X04.01 23-AUG-82 12:02:53 PAGE 145-5 SEQ 1073
SYMBOL TABLE

DYNAMIC MEMORY AVAILABLE FOR 71 PAGES
,B:UDAT4.L50/C=$DMACRO,[30,27]UDAT4

CROSS REFERENCE TABLE (CREF V04.00)

```
AFTOP   65-2#    74-68    75-145   85-17    86-65
AFTWRT  62-39    99-129   101-19#  102-19
ALLOCM  96-14    96-17    96-40    97-21    98-2#
AREA0   62-9     62-57#   63-5     63-5     95-5     95-5     99-5     99-5     101-5    101-5    102-5    102-5    105-5    105-5
        106-5    106-5    107-5    107-5    108-5    108-5    109-5    109-5    110-5    110-5    111-5    111-5    112-5    112-5
        122-5    122-5
AREA1   62-11    63-127#  64-5     64-5
AREA2   62-13    72-20#   73-5     73-5     75-5     75-5     82-5     82-5     86-5     86-5
AREAI   62-17    62-67#   123-5    123-5    124-5    124-5    125-5    125-5    126-5    126-5    131-5    131-5    138-5    138-5
        143-5    143-5    145-5    145-5
ATTN     8-23#   35-18    51-49    53-18    53-25    53-27    55-18    55-77    56-11    62-67    120-89
AVAIL    8-26#   35-18    51-49    53-16    53-23    55-41    62-67    120-82   121-56
BBLOP   62-67    62-67#
BEEXT   75-86    75-138   75-140#
BESDWN  75-53    75-76#
BESUP   75-119   75-139#
BEUSED  10-52#   62-67    62-67    62-67    64-36    64-42    126-42   131-97   135-17   135-43   135-61   138-59
BF.DAT   5-88#
BF.ECC   5-90#
BF.EDC   5-89#   99-114*  106-85   106-112  107-71   107-97   143-179*
BLDBB   62-67    62-67#
BLDUNT  62-67    62-67#
BLKCHK  42-2#    101-89   105-58   106-60   110-93
BREAK    6-3#
BSUSEX  62-67    62-67#
BUFARA  62-15    62-67    62-67    62-67    98-15    100-57   101-186  104-29   105-173  106-138  107-128  108-72   109-89   110-131
        111-76   116-25   117-5    117-5    120-5    120-5    121-5    121-5
BUFFLG   5-62#    5-71    101-65   104-13   105-56
BUFSIZ  37-15    37-20    38-41#
BUILDP  62-37    95-19#   99-19    108-58   109-81   117-86   120-75
BULDUM  44-2#    99-60    102-68   143-49
C2DFTL   8-65#   40-96    124-38   125-39   131-151
C2HARD   8-64#   106-123  107-106
CALC    45-2#    81-17    81-29    95-45    115-25   118-16
CALCSC  45-60    46-3#    143-67
CBB2    62-67#   129-12   135-13   135-28   135-39   135-57   139-14   139-26   139-40
CHAINS  46-37#   95-44*   96-7     96-18*   99-79    100-7    101-49   103-6    104-9    105-42   106-41   106-104  107-39   110-40
        110-96   111-37   111-53*  114-6
CHGMOD   8-34#   41-[illegible]
CHKBB   138-54   139-2#
CHKBES  131-104  135-2#
CHKCYL  131-101  133-2#
CHKDN    50-121   50-125   58-2#
CHKECC  62-43    106-71   107-19#  108-19
CHKEDC  62-42    105-165  106-19#  107-19
CHKTG   138-61   140-2#
CHKUP    50-79    50-118   50-128   59-2#
CHNMAX  46-39#   62-67*   63-103   63-105
CHRRES   8-45#   62-67
CLCLBN  74-27    74-33#
CLCMAX  131-94   137-2#
CMP2    42-15    43-2#    75-52    76-13
CMPDAT  62-46    106-102  106-127  107-119  109-60   110-19#  111-19
CMPEDC  39-2#    99-113   106-83   107-70   143-178
CMPERR  110-73   110-82   110-90#
CMXEX   110-88   110-95   110-125#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
CNTLOP  73-37#   73-41
COMCHR  62-59    123-40   124-19#  125-19
COMPAR   6-9#
COMPDP 141-10    141-31   141-49   142-2#
COMPLT   8-43#    41-6     41-9     41-10    41-11    41-12    41-13    41-15    62-67
COMPSC 131-93    136-2#
COPBB   62-67     62-67#
COPFIN  99-99     99-107#
COPLP0  99-92#    99-102
COPLP1  99-96#    99-101
COPLP2  99-104#   99-106
COUNTD  73-38     73-42#
CPYBEX  62-67     62-67#
CR.CLR  41-9#     57-52
CR.DIS  41-13#    48-40
CR.ERR  41-12#   109-66
CR.GCR  62-67#   124-33
CR.GST  41-8#     55-44    57-15    62-67
CR.INR  41-15#   121-39   121-58   143-73
CR.MOD  41-10#    57-62
CR.ONL  41-6#     57-9
CR.RUN  62-67#   125-32
CR.SCR  62-67#   131-52
CR.SEK  41-11#   119-9    143-80
CSCEXT 110-76    110-86#
CURBN   46-5#     81-13*   81-15*   81-57    81-59    81-72*   81-74*   81-76*   81-78*   95-37*   95-38*   95-40*   95-41*   97-42
        97-44     97-73*   97-75*  112-74*  112-78*  113-17   113-17   114-14   114-16*  114-51   114-51   115-22*  115-23*  115-41
       115-41    118-10*  118-11*  118-13*  118-14*  143-65*  143-66*  143-124  143-133  143-187* 143-189*
CVT     6-19#    45-62    143-69
CYL     46-8#    81-19    81-31    95-46*   95-48*   115-27*  115-30*  117-60   117-60   118-17   119-6    119-7    119-24   143-77
       143-78
CYLBN  133-11    133-24   134-2#
CYLSCR  81-18    81-30    81-82#
D$      62-84#  131-82   135-65   135-66
D.LIMT  10-4#    44-15*
D.SCHR  10-5#    44-18*
DATCMP  10-57#   62-67    65-14    69-6     69-17    106-98   106-125  107-117  109-58
DATPRE   7-32#  100-10
DBNCYL   7-41#  137-10   137-18
DCEXT   69-7     69-11    69-16    69-19#
DCLOCK   8-24#   35-33
DCLR    41-25#   57-51*
DCMPAL  10-59#   62-67    69-12
DCMPYS  69-13    69-17#
DCREAD  69-9     69-12#
DCYLS   10-44#   44-8     45-17    62-67    62-67    97-49    101-122  105-88   131-80   133-35   135-36   139-22   143-205
DEBUG    2-50#    3-2     48-3     51-55    52-6     62-67    62-67    62-84    63-35    63-47    99-32    101-33   102-32   105-35
       106-35   107-32   109-32   110-32   112-30   117-36   120-34   121-32
DIE     10-10#   62-67   124-40   125-41   131-121  131-153  138-68   143-211  145-73
DINIT    6-12#   60-56
DIREC   10-17#   50-14    50-116   58-23    59-35    75-33    81-54    81-70    86-35
DIS     41-13    41-20#
DISCON   8-32#   41-20
DONE     8-18#   48-52
DOWNG   86-41    90-2#
DOWNT   86-39    88-2#    90-23
```

CROSS REFERENCE TABLE (CREF V04.00)

```
DRC      41-9     41-24#
DRINIT   10-32#   55-28    55-30    60-58
DROP     10-15#   48-30    48-39    50-28    50-84    50-109   50-132   55-6     56-7     56-35    56-43    58-17    59-18    122-59
         131-126  131-142  138-73   138-89
DRPALL   56-37    62-52    122-19#  123-19
DRTYPE    7-15#
DRVCLR    8-37#   41-24
DRVEXT  145-74    145-89#
DRVID     7-14#   124-42   124-43   124-44
DRVONL    8-35#   41-34
DRVRUN    8-36#   62-67
DSABLE   56-32    60-12#   99-39    102-39   109-47   117-84   120-96   143-230
DSERNM   62-67#   124-42*  124-43*  124-44*  143-213  143-214  143-215
DUMSDI   44-15*   44-18*   46-14#   46-28    46-30    97-38    143-137
DUPPKT    8-3#     8-4      8-5      8-6      8-7      8-8      8-9      8-10     8-11     8-12     8-13     8-14     8-15     8-16
          8-17     8-18
ECC       6-16#   107-49   143-169
ECCCHK   10-45#   62-67    106-69
ECCFLG    5-63#   106-55   143-158
ECLRSH    7-11#   124-55
ECHO      6-11#
ECHOC     8-48#
EDCLOP   39-10#   39-16
ENABLE   60-3#    60-25    60-34    60-43    63-115   108-32   113-30   115-44   120-47
EOC       5-64#    5-68     5-71    96-33    96-41    101-70   104-17   105-161  111-51
ER1      62-84#   120-92
ER10     62-84#   107-92
ER1000   62-67    62-84#
ER11     62-84#   109-48
ER12     62-84#   110-97
ER13     53-20    62-84#
ER14     62-84#   117-59
ER15     62-84#   120-71
ER16     62-67    62-84#
ER17     35-6     62-84#
ER17A    35-8     62-84#
ER18     62-84#   100-44
ER19     62-84#   105-96
ER2      53-28    62-84#
ER20     62-84#   105-80
ER21     62-84#   143-207
ER23     57-25    62-84#
ER24     62-84#   106-107
ER25     62-84#    99-50
ER26     62-84#   102-53
ER28     62-84#   138-40
ER3      62-84#   120-108
ER33     51-23    62-84#
ER34     62-84#   105-114
ER35     62-84#   105-132
ER36     62-84#   105-118
ER37     62-84#   105-128
ER38     62-84#   105-137
ER4      62-84#   11?-34
ER40     62-84#   113-10
ER41     62-84#   114-44
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

	128-18	129-27	130-14	131-84	131-90	136-9	136-16	137-12	137-19	140-37	140-41	141-9	141-25	141-46
	141-47	143-54	143-56											
HICYL	7-23#	45-37	45-49	115-27										
HIDBN	7-29#	45-53												
HILBN	7-25#	45-32												
HIMEM	5-12#	9-79	46-26	62-67										
HIRBN	7-28#	45-28												
HISEED	46-24#	71-9	71-19	71-25*	74-23	82-40								
HIXBN	7-26#	143-126												
HOSTRQ	36-7	37-2#	40-106	48-33	50-77	51-26	54-6	56-4	56-13	62-67	62-67	62-67	62-67	62-67
	62-67	62-67	62-67	62-67	100-51	101-144	103-30	105-122	106-92	107-86	107-89	117-68	120-63	120-70
	120-91	122-57	131-120	138-47	138-67	143-102	143-210	143-221						
HRDREV	7-13#													
HRQ.01	35-10*	36-7*	38-6#	40-24*	40-40*	40-56*	40-60*	40-64*	40-68*	40-72*	40-75*	40-82*	40-106*	50-77*
	50-77*	51-23*	53-20*	53-28*	54-3*	54-4*	54-7*	56-5*	56-13*	57-25*	62-67	62-67	62-67	62-67
	62-67	62-67*	62-67*	62-67*	62-67*	62-67*	62-67*	62-67*	62-67*	62-67*	99-50*	100-15*	100-44*	100-51*
	101-114*	101-129*	101-144*	101-145*	101-150*	101-154*	101-159*	101-164*	101-166*	102-53*	103-11*	103-30*	105-80*	105-96*
	105-114*	105-118*	105-122*	105-123*	105-128*	105-132*	105-137*	105-142*	105-144*	106-73*	106-92*	106-107*	107-53*	107-59*
	107-84*	107-89*	107-89*	107-92*	109-48*	110-97*	113-10*	114-44*	115-34*	117-59*	117-68*	120-60*	120-61*	120-70*
	120-71*	120-91*	120-92*	120-108*	122-49*	134-22*	135-21*	135-23*	135-30*	135-53*	135-67*	135-69*	138-40*	138-47*
	138-47*	139-18*	139-36*	139-42*	140-45*	141-29*	143-100*	143-207*	143-212*	145-71*				
HRQ.02	35-9*	36-7*	38-7#	40-24*	40-40*	40-56*	40-60*	40-64*	40-68*	40-72*	40-75*	40-82*	40-96*	40-106*
	50-77*	51-23*	53-20*	53-28*	55-1	56-13*	57-25*	62-67	62-67	62-67*	62-67*	99-50*	99-123*	100-15*
	100-44*	100-51*	101-114*	101-129*	101-144*	101-145*	101-150*	101-154*	101-159*	101-164*	101-166*	102-53*	103-11*	103-30*
	105-80*	105-96*	105-114*	105-118*	105-122*	105-123*	105-128*	105-132*	105-137*	105-142*	105-144*	106-73*	106-92*	106-107*
	106-123*	107-53*	107-59*	107-77*	107-80*	107-89*	107-92*	107-106*	109-48*	110-97*	111-64*	113-10*	114-44*	115-34*
	117-59*	117-68*	120-70*	120-71*	120-91*	120-92*	120-108*	122-37*	124-38*	125-39*	131-151*	134-22*	135-21*	135-23*
	135-30*	135-53*	135-67*	135-69*	138-40*	138-47*	139-18*	139-36*	139-42*	140-45*	141-29*	143-97*	143-207*	143-213*
	145-71*													
HRQ.03	36-7*	38-8#	40-106*	51-24*	56-1*	56-2*	56-13*	62-67	62-67	62-67	62-67*	62-67*	99-124*	100-51*
	101-144*	103-30*	105-122*	106-92*	107-73*	107-89*	111-65*	117-68*	120-70*	120-91*	122-55*	131-117*	131-118*	138-64*
	138-65*	143-98*	143-208*	143-214*										
HRQ.04	35-6*	35-8*	36-7*	38-9#	40-24*	40-40*	40-56*	40-60*	40-64*	40-68*	40-72*	40-75*	40-82*	40-106*
	51-23*	53-20*	53-28*	56-13*	57-25*	62-67*	62-67*	99-50*	100-15*	100-44*	100-51*	101-114*	101-129*	101-144*
	101-145*	101-150*	101-154*	101-159*	101-164*	101-166*	102-53*	103-11*	103-30*	105-80*	105-96*	105-114*	105-118*	105-122*
	105-123*	105-128*	105-132*	105-137*	105-142*	105-144*	106-73*	106-92*	106-107*	107-53*	107-59*	107-89*	107-92*	109-48*
	110-97*	113-10*	114-44*	115-34*	117-59*	117-68*	120-70*	120-71*	120-91*	120-92*	120-108*	122-38	134-22*	135-21*
	135-23*	135-30*	135-53*	135-67*	135-69*	138-40*	139-18*	139-36*	139-42*	140-45*	141-29*	143-99*	143-207*	143-215*
	145-71*													
HRQ.05	38-10#	51-23*	53-29*	57-13*	57-19*	57-28*	57-32*	57-34*	57-56*	57-64*	62-67	62-67*	99-50*	100-15*
	100-44*	101-130*	101-166*	101-169*	102-53*	103-11*	105-97*	105-144*	105-147*	106-73*	106-107*	107-60*	107-89*	107-92*
	109-48*	109-70*	110-98*	110-101*	113-14*	113-16*	114-48*	114-50*	115-38*	115-40*	117-59*	119-33*	120-71*	120-92*
	120-108*	121-65*	124-37*	125-38*	131-150*	134-22*	135-21*	135-23*	135-30*	135-53*	135-67*	135-69*	139-18*	139-36*
	139-42*	140-45*	141-29*	143-207*	143-216*	145-71*								
HRQ.06	38-11#	40-73*	40-76*	40-83*	51-23*	57-35*	99-50*	100-15*	100-45*	101-130*	101-166*	101-170*	102-53*	103-11*
	105-97*	105-144*	105-148*	106-73*	106-107*	107-60*	107-89*	107-92*	109-48*	110-102*	113-17*	114-51*	115-41*	117-59*
	120-71*	120-92*	120-108*	135-47*	135-53*	135-65*	135-69*	139-36*	139-42*	140-45*	141-29*	143-217*		
HRQ.07	38-12#	40-77*	40-83*	62-67	62-67	99-50*	100-15*	101-130*	101-167*	101-170*	102-53*	103-11*	105-97*	105-145*
	105-148*	106-73*	106-107*	107-60*	107-89*	107-92*	109-48*	110-102*	113-17*	114-51*	115-41*	117-59*	120-71*	120-92*
	120-108*	135-48*	135-53*	135-66*	135-69*	139-36*	139-42*	140-45*	143-218*					
HRQ.08	38-13#	40-83*	99-51*	100-16*	101-131*	101-170*	102-54*	103-12*	105-98*	105-148*	106-74*	106-108*	107-61*	107-89*
	107-93*	109-49*	110-102*	113-17*	115-41*	117-60*	120-71*	120-92*	120-109*	143-219*				
HRQ.09	38-14#	99-51*	101-131*	101-170*	102-54*	105-98*	105-148*	106-74*	106-108*	107-61*	107-89*	107-93*	109-49*	110-103*
	113-17*	115-41*	117-60*	120-72*	120-92*	120-109*								
HRQ.10	38-15#	99-51*	101-131*	101-171*	102-54*	105-98*	105-149*	106-74*	106-108*	107-61*	107-89*	107-93*	109-49*	110-103*
	117-60*	120-72*	120-93*	120-109*										

CROSS REFERENCE TABLE (CREF V04.00)

```
HRQ.11   38-16#   99-51*  101-132* 101-171* 102-54*  105-99*  105-149* 106-75*  106-109* 107-62*  107-94*  109-49*  110-104* 117-60*
        120-72*  120-94*  120-110*
HRQ.12   38-17#  101-132* 101-171* 105-99*  105-149* 106-75*  106-109* 107-62*  107-94*  109-49*  110-104* 117-60*  120-72*
HRQ.13   38-18#  101-133* 101-172* 105-100* 105-150* 106-76*  106-110* 107-63*  107-95*  109-50*  110-104* 117-60*  120-72*
HRQ.14   38-19#  101-133* 101-172* 105-100* 105-150* 106-76*  106-110* 107-63*  107-95*  109-50*  110-105* 117-60*  120-72*
HRQ.15   38-20#  101-133* 101-173* 105-100* 105-151* 106-76*  106-110* 107-63*  107-95*  110-105* 120-73*
HRQ.16   38-21#  101-133* 101-173* 105-100* 105-151* 106-77*  106-111* 107-64*  107-96*  110-105*
HRQ.17   38-22#  101-134* 101-173* 105-101* 105-151* 106-77*  106-111* 107-64*  107-96*  110-105*
HRQ.18   38-23#  101-134* 101-173* 105-101* 105-151* 106-77*  106-111* 107-64*  107-96*  110-122*
HRQ.19   38-24#  101-134* 101-174* 105-101* 105-152* 106-78*  106-112* 107-65*  107-97*  110-122*
HRQ.20   38-25#  101-174* 105-152* 106-112* 107-97*  110-122*
HRQ.21   38-26#  101-174* 105-152* 106-112* 107-97*  110-122*
HRQ.22   38-27#  101-175* 105-153* 110-122*
HRQ.23   38-28#  110-122*
HRQ.24   38-29#  110-122*
HRQ.25   38-30#  110-122*
HRQ.26   38-31#  110-122*
HRQ.27   38-32#  110-122*
HRQ.28   38-33#  110-122*
HRQ.29   38-34#  110-123*
HRQ.30   38-35#  110-123*
HRQ.31   38-36#  110-123*
HRQ.32   38-37#  110-123*
HRQ.33   38-38#
HRQ.34   38-39#
HRQ.RQ   35-11*   37-13*   37-14    37-19    37-22*   38-5#    38-41    40-24*   40-40*   40-56*   40-60*   40-64*   40-68*   40-72*
         40-75*   40-82*   51-23*   51-25    53-20*   53-28*   54-1*    54-5     55-10    56-3     57-25*   62-67    62-67    62-67*
         62-67*   99-50*   99-126*  100-15*  100-44*  101-114* 101-129* 101-145* 101-150* 101-154* 101-159* 101-164* 101-166* 102-53*
        103-11*  105-80*  105-96*  105-114* 105-118* 105-123* 105-128* 105-132* 105-137* 105-142* 105-144* 106-73*  106-107* 106-124*
        107-53*  107-59*  107-92*  107-107* 109-48*  110-97*  111-67*  113-10*  114-44*  115-34*  117-59*  120-71*  120-92*  120-108*
        124-39*  125-40*  131-119  131-152* 134-22*  135-21*  135-23*  135-30*  135-53*  135-67*  135-69*  138-40*  138-66   139-18*
        139-36*  139-42*  140-45*  141-29*  143-207* 143-209  145-71*
ID       81-27    81-47    81-69#
INCODT    2-51#    2-68    34-21    60-66
INDEX    46-12#   97-57
INITD    62-65   143-229  145-19#
INITW    10-16#   48-34    50-32    50-74    50-85    59-22    59-32    62-67    62-67    63-72    64-34    66-8     68-10    89-11
        138-42   138-44   145-45   145-51
INR      41-15    41-27#
INS      41-11    41-28#  119-6*   119-7*   119-8*
INSEEK    8-42#   41-28
INSET    62-58    62-67   123-19#  124-19
INTEDC    9-77#   39-9
INTINP   10-11#   56-38    62-67    62-67
IRECLB    8-41#   41-27
ISUEXT  119-32   119-34#
ISUSEK  117-72   119-2#
IWIPRG   10-9#    48-33    48-49    48-50    50-31    50-73    59-21    59-31    77-9     81-11    86-53    145-53
JMPRET   51-52    52-4#    63-122   64-49    70-10    74-71    75-148   85-20    86-68    97-87    99-131   101-181  102-72   105-168
        106-133  107-123  108-67   109-85   110-127  111-71   112-89   117-89   120-112  121-77   123-41   124-84   125-43   126-55
        131-157  138-100  143-231  145-89
L$       62-84#  116-18   131-86   135-47   135-48
L2.EOF    8-56#   40-90    40-101  121-58
L2.ERS    8-55#    8-56    40-80    40-83
L2.OPC    8-52#    8-53    40-14    55-47    62-67
L2.RLN    8-54#    8-55    40-43    55-57    62-67
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
        107-60   107-60   107-60   107-60   107-60   107-60   107-60#  107-60#  107-60#  107-61   107-61   107-61   107-61   107-61
        107-61   107-61#  107-61#  107-61#  107-62   107-62   107-62   107-62   107-62#  107-62#  107-63   107-63   107-63   107-63
        107-63   107-63   107-63#  107-63#  107-63#  107-64   107-64   107-64   107-64   107-64   107-64   107-64#  107-64#  107-64#
        107-65   107-65   107-65#  107-89   107-89   107-89   107-89   107-89   107-89   107-89   107-89   107-89   107-89   107-89
        107-89   107-89   107-89   107-89   107-89   107-89#  107-89#  107-89#  107-89#  107-89#  107-89#  107-89#  107-89#  107-89#
        107-92   107-92   107-92   107-92   107-92   107-92   107-92#  107-92#  107-92#  107-92#  107-93   107-93   107-93   107-93
        107-93   107-93   107-93#  107-93#  107-93#  107-94   107-94   107-94   107-94   107-94#  107-94#  107-95   107-95   107-95
        107-95   107-95   107-95   107-95#  107-95#  107-95#  107-96   107-96   107-96   107-96   107-96   107-96   107-96#  107-96#
        107-96#  107-97   107-97   107-97   107-97   107-97   107-97   107-97#  107-97#  107-97#  109-48   109-48   109-48   109-48
        109-48   109-48   109-48#  109-48#  109-48#  109-48#  109-49   109-49   109-49   109-49   109-49   109-49   109-49   109-49
        109-49   109-49   109-49#  109-49#  109-49#  109-49#  109-49#  109-50   109-50   109-50   109-50   109-50#  109-50#  109-70
        109-70   109-70#  109-70#  110-97#  110-98   110-98   110-98#  110-98#  110-101  110-101  110-101#  110-101#  110-102  110-102
        110-102  110-102  110-102  110-102  110-102#  110-102#  110-102#  110-103  110-103  110-103  110-103  110-103#  110-103#  110-104
        110-104  110-104  110-104  110-104  110-104  110-104#  110-104#  110-104#  110-105  110-105  110-105  110-105  110-105  110-105
        110-105  110-105  110-105#  110-105#  110-105#  110-105#  110-122  110-122  110-122  110-122  110-122  110-122  110-122  110-122
        110-122  110-122  110-122  110-122  110-122  110-122  110-122  110-122  110-122  110-122  110-122  110-122  110-122  110-122
        110-122# 110-122# 110-122# 110-122# 110-122# 110-122# 110-122# 110-122# 110-122# 110-122# 110-122# 110-123  110-123  110-123
        110-123  110-123  110-123  110-123  110-123  110-123#  110-123#  110-123#  110-123#  113-10#  113-14   113-14   113-14#  113-14#
        113-16   113-16   113-16#  113-16#  113-17   113-17   113-17   113-17   113-17   113-17   113-17   113-17   113-17#  113-17#
        113-17#  113-17#  114-44#  114-48   114-48   114-48#  114-48#  114-50   114-50   114-50#  114-50#  114-51   114-51   114-51
        114-51   114-51#  114-51#  115-34#  115-38   115-38   115-38#  115-38#  115-40   115-40   115-40#  115-40#  115-41   115-41
        115-41   115-41   115-41   115-41   115-41   115-41   115-41#  115-41#  115-41#  115-41#  117-59   117-59   117-59   117-59
        117-59   117-59   117-59#  117-59#  117-59#  117-59#  117-60   117-60   117-60   117-60   117-60   117-60   117-60   117-60
        117-60   117-60   117-60   117-60   117-60   117-60   117-60#  117-60#  117-60#  117-60#  117-60   117-60#  117-60#  119-33
        119-33   119-33#  119-33#  120-71   120-71   120-71   120-71   120-71   120-71   120-71   120-71   120-71#  120-71#  120-71#
        120-71#  120-71#  120-72   120-72   120-72   120-72   120-72   120-72   120-72   120-72   120-72   120-72   120-72   120-72
        120-72#  120-72#  120-72#  120-72#  120-72#  120-72#  120-73   120-73   120-73#  120-92   120-92   120-92   120-92   120-92
        120-92   120-92   120-92   120-92   120-92   120-92#  120-92#  120-92#  120-92#  120-92#  120-92#  120-93   120-93   120-93#
        120-94   120-108  120-108  120-108  120-108  120-108  120-108  120-108  120-108#  120-108#  120-108#  120-108#  120-109  120-109  120-109
        120-109  120-109  120-109  120-109#  120-109#  120-109#  120-110  121-65   121-65   121-65#  121-65#  124-37   124-37   124-37#
        124-37#  125-38   125-38   125-38#  125-38#  131-150  131-150  131-150#  131-150#  134-22   134-22   134-22#  134-22#  135-21
        135-21   135-21#  135-21#  135-23   135-23   135-23#  135-23#  135-30   135-30   135-30#  135-30#  135-47   135-47   135-47#
        135-47#  135-48   135-48   135-48#  135-48#  135-53   135-53   135-53   135-53   135-53   135-53   135-53#  135-53#  135-53#
        135-53#  135-65   135-65   135-65#  135-65#  135-66   135-66   135-66#  135-66#  135-67   135-67   135-67#  135-67#  135-69
        135-69   135-69   135-69   135-69   135-69   135-69#  135-69#  135-69#  135-69#  138-40#  138-47#  139-18   139-18   139-18#
        139-18#  139-36   139-36   139-36   139-36   139-36   139-36   139-36#  139-36#  139-36#  139-36#  139-42   139-42   139-42
        139-42   139-42   139-42   139-42#  139-42#  139-42#  139-42#  140-45   140-45   140-45   140-45   140-45   140-45   140-45#
        140-45#  140-45#  140-45#  141-29   141-29   141-29   141-29   141-29#  141-29#  141-29#  143-207  143-207  143-207#  143-207#
        145-71   145-71   145-71#  145-71#
NXTLEV   10-34#   63-116   63-119   108-63   109-40   109-73   109-75   120-77
NXTRCT  112-88   115-2#
NXTTRK   77-12   78-31   79-32   81-2#
OCNT$    62-29#  62-30   62-30   62-30#  62-31   62-31   62-31#  62-32   62-32   62-32#  62-33   62-33   62-33#  62-34
         62-34   62-34#  62-35   62-35   62-35#  62-36   62-36   62-36#  62-37   62-37   62-37#  62-38   62-38   62-38#
         62-39   62-39   62-39#  62-40   62-40   62-40#  62-41   62-41   62-41#  62-42   62-42   62-42#  62-43   62-43
         62-43#  62-44   62-44   62-44#  62-45   62-45   62-45#  62-46   62-46   62-46#  62-47   62-47   62-47#  62-48
         62-48   62-48#  62-49   62-49   62-49#  62-50   62-50   62-50#  62-51   62-51   62-51#  62-52   62-52   62-52#
         62-58   62-58   62-58#  62-59   62-59   62-59#  62-60   62-60   62-60#  62-61   62-61   62-61#  62-62   62-62
         62-62#  62-63   62-63   62-63#  62-64   62-64   62-64#  62-65   62-65   62-65#
ONEPAT   99-95   99-103#
ONEPAX  110-70   110-80#
ONL      41-6    41-34#
ONLYCL   10-49#  62-67   131-99   135-19   135-45   135-63
OTABLE   50-149  51-8    51-11    51-15    51-30    62-30#  62-66   62-67   62-67*
OUTO     76-14   76-16#
OV...     2-77    2-77    2-77#   62-84    62-84    62-84    62-84#  63-5    63-5    63-5    63-5#   64-5    64-5    64-5
```

CROSS REFERENCE TABLE (CREF V04.00)

	64-5#	73-5	73-5	73-5	73-5#	75-5	75-5	75-5	75-5#	82-5	82-5	82-5	82-5#	86-5
	86-5	86-5	86-5#	95-5	95-5	95-5	95-5#	99-5	99-5	99-5	99-5#	101-5	101-5	101-5
	101-5#	102-5	102-5	102-5	102-5#	105-5	105-5	105-5	105-5#	106-5	106-5	106-5	106-5#	107-5
	107-5	107-5	107-5#	108-5	108-5	108-5	108-5#	109-5	109-5	109-5	109-5#	110-5	110-5	110-5
	110-5#	111-5	111-5	111-5	111-5#	112-5	112-5	112-5	112-5#	117-5	117-5	117-5	117-5#	120-5
	120-5	120-5	120-5#	121-5	121-5	121-5	121-5#	122-5	122-5	122-5	122-5#	123-5	123-5	123-5
	123-5#	124-5	124-5	124-5	124-5#	125-5	125-5	125-5	125-5#	126-5	126-5	126-5	126-5#	131-5
	131-5	131-5	131-5#	138-5	138-5	138-5	138-5#	143-5	143-5	143-5	143-5#	145-5	145-5	145-5
	145-5#	145-113	145-113#											
OVE.AW	101-5#	102-5												
OVE.BP	95-5#	99-5												
OVE.CC	124-5#	125-5												
OVE.CD	110-5#	111-5												
OVE.CK	107-5#	108-5												
OVE.DA	122-5#	123-5												
OVE.EC	108-5#	109-5												
OVE.ED	106-5#	107-5												
OVE.GS	143-5#	145-5												
OVE.ID	145-5#	145-113												
OVE.IN	123-5#	124-5												
OVE.LM	111-5#	112-5												
OVE.MN	2-77#	62-84												
OVE.MS	62-84#	63-5												
OVE.NL	109-5#	110-5												
OVE.NO	64-5#	73-5												
OVE.R	102-5#	105-5												
OVE.RB	73-5#	75-5												
OVE.RC	121-5#	122-5												
OVE.RT	82-5#	86-5												
OVE.RV	112-5#	117-5												
OVE.SO	131-5#	138-5												
OVE.S1	138-5#	143-5												
OVE.SB	75-5#	82-5												
OVE.SC	105-5#	106-5												
OVE.SK	117-5#	120-5												
OVE.SO	126-5#	131-5												
OVE.SP	125-5#	126-5												
OVE.ST	86-5#	95-5												
OVE.SU	63-5#	64-5												
OVE.TS	120-5#	121-5												
OVE.W	99-5#	101-5												
OVL...	2-77	2-77#	62-84	62-84#	63-5	63-5#	64-5	64-5#	73-5	73-5#	75-5	75-5#	82-5	82-5#
	86-5	86-5#	95-5	95-5#	99-5	99-5#	101-5	101-5#	102-5	102-5#	105-5	105-5#	106-5	106-5#
	107-5	107-5#	108-5	108-5#	109-5	109-5#	110-5	110-5#	111-5	111-5#	112-5	112-5#	117-5	117-5#
	120-5	120-5#	121-5	121-5#	122-5	122-5#	123-5	123-5#	124-5	124-5#	125-5	125-5#	126-5	126-5#
	131-5	131-5#	138-5	138-5#	143-5	143-5#	145-5	145-5#	145-113	145-113#				
OVL.AW	62-39	102-5	102-5	102-5#										
OVL.BP	62-37	99-5	99-5	99-5#										
OVL.CC	62-59	125-5	125-5	125-5#										
OVL.CD	62-46	111-5	111-5	111-5#										
OVL.CK	62-43	108-5	108-5	108-5#										
OVL.DA	62-52	123-5	123-5	123-5#										
OVL.EC	62-44	109-5	109-5	109-5#										
OVL.ED	62-42	107-5	107-5	107-5#										
OVL.GS	62-64	145-5	145-5	145-5#										
OVL.ID	62-65	145-113	145-113	145-113#										

CROSS REFERENCE TABLE (CREF V04.00)

```
OVL.IN  62-58   124-5    124-5    124-5#
OVL.LM  62-47   112-5    112-5    112-5#
OVL.MN   2-77     2-77    62-84    62-84    62-84#
OVL.MS  62-30    63-5     63-5     63-5#
OVL.NL  62-45   110-5    110-5    110-5#
OVL.NO  62-32    73-5     73-5     73-5#
OVL.R   62-40   105-5    105-5    105-5#
OVL.RB  62-33    75-5     75-5     75-5#
OVL.RC  62-51   122-5    122-5    122-5#
OVL.RT  62-35    86-5     86-5     86-5#
OVL.RV  62-48   117-5    117-5    117-5#
OVL.SO  62-62   138-5    138-5    138-5#
OVL.S1  62-63   143-5    143-5    143-5#
OVL.SB  62-34    82-5     82-5     82-5#
OVL.SC  62-41   106-5    106-5    106-5#
OVL.SK  62-49   120-5    120-5    120-5#
OVL.SO  62-61   131-5    131-5    131-5#
OVL.SP  62-60   126-5    126-5    126-5#
OVL.ST  62-36    95-5     95-5     95-5#
OVL.SU  62-31    64-5     64-5     64-5#
OVL.TS  62-50   121-5    121-5    121-5#
OVL.W   62-38   101-5    101-5    101-5#
OVS.AW 101-5#
OVS.BP  95-5#
OVS.CC 124-5#
OVS.CD 110-5#
OVS.CK 107-5#
OVS.DA 122-5#
OVS.EC 108-5#
OVS.ED 106-5#
OVS.GS 143-5#
OVS.ID 145-5#
OVS.IN 123-5#
OVS.LM 111-5#
OVS.MN   2-77#
OVS.MS  62-84#
OVS.NL 109-5#
OVS.NO  64-5#
OVS.R  102-5#
OVS.RB  73-5#
OVS.RC 121-5#
OVS.RT  82-5#
OVS.RV 112-5#
OVS.SO 131-5#
OVS.S1 138-5#
OVS.SB  75-5#
OVS.SC 105-5#
OVS.SK 117-5#
OVS.SO 126-5#
OVS.SP 125-5#
OVS.ST  86-5#
OVS.SU  63-5#
OVS.TS 120-5#
OVS.W   99-5#
OVSTRT   5-13#   62-67    62-67
PARTO   62-8#    62-30    62-30    62-31    62-31    62-31    62-32    62-33    62-34    62-35    62-36    62-37    62-37    62-37
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
REVEC   10-21#   50-14    65-14    96-11    97-9     97-82    102-51   102-63   105-90   106-58   106-94   106-115  107-104  107-113
        109-54   112-38   112-46   116-7
REVS     7-16#
REVSOK  112-81   113-2#
RM       7-31#
RNDO     67-9     67-12#   67-14
RNDBE    62-33    64-38    73-19#   75-19
RNDTG    62-35    64-40    82-19#   86-19
RONLY    10-46#   62-67    62-67    66-10    138-36   145-75
ROOT     48-24#   48-51    62-67
ROOTLP   48-26#   48-43    48-46
RREAL     5-67#   97-16    143-136
RSTOP     5-68#   97-18    143-115
RTDS      35-2#   40-32    51-46    53-13    57-6     120-41   121-51
RTDSL     35-2    35-16#   55-15    55-38    60-53    143-91
RTRIES   10-48#   62-67    99-48    102-48   106-119  107-101
RUN      62-67    62-67#
RVFAIL  113-19   114-27   114-53   115-33   116-2#
RW.ANG    5-58#   97-41*   97-62*   101-132  101-172  105-99   105-150  106-76   106-110  107-63   107-89   107-95   110-104
RW.BUF    5-53#    5-54    97-37*   99-80    106-82   106-106  107-69   110-41   114-7
RW.CMD    5-56#    5-57    97-12*   97-17*   101-133  101-173  105-100  105-151  106-77   106-111  107-64   107-89   107-96   110-105
        143-135* 143-136*
RW.HI     5-55#    5-56    97-56*   101-51*  101-130  101-170  104-11*  105-97   105-148  106-74   106-108  107-61   107-89   107-93
        110-102  143-134*
RW.LOW    5-54#    5-55    42-13    97-43*   101-50*  101-130  101-170  104-10*  105-97   105-148  106-74   106-108  107-61   107-89
        107-93   110-102  143-124*
RW.SDI    5-57#    5-58    97-39*   143-137*
RW.STA    5-43#    5-53    96-24    96-47    97-28    101-56   105-47   106-46   111-41   143-115* 143-158*
RWRDY     8-28#   35-18    120-50   143-96
S.BADP    9-64#    9-65    42-11    62-67*   62-67*   129-6    139-8
S.BESS    9-65#    9-70    62-67    62-67    62-67    62-67    73-35    73-56    75-37    75-89    76-7     127-6    127-8    133-8
        133-9    133-14*  133-16*  133-17   133-20   133-30*  133-31*  133-39*  133-43*  133-46*  133-48*  135-9    135-11   141-57
        141-59   141-60   141-64   141-70
S.LETR    9-59#    9-60    99-50    101-130  101-170  102-53   105-97   105-148  106-74   106-108  107-61   107-89   107-93   109-48
        110-102  112-49*  116-19*  117-59   120-71   131-88*
S.MCNT    9-70#    9-71    82-37    82-42    82-45    82-48    91-7     91-9     92-10    92-12
S.MEGR    9-62#    9-63    105-64   105-65*  105-158  105-160* 106-62   106-64*  108-39   108-40*  111-60   111-68*  131-58*
S.MEGW    9-63#    9-64    99-118   99-122*  101-79   101-80*  131-59*
S.PARM    9-55#    9-56    44-7     45-16    50-26    50-62    50-76    50-104   56-42    56-45    58-15    59-16    62-67    62-67
         63-69    64-33    66-7     84-7     86-34    86-50    97-48    99-47    101-121  102-47   105-87   106-68   106-118  107-100
        126-41   131-79   131-96   131-128  131-140  133-34   135-16   135-35   135-42   135-60   138-35   138-46   138-58   138-75
        138-87   139-21   140-28   141-38   142-9    143-204  145-39   145-63
S.PAT     9-58#    9-59    62-67*   67-8
S.SCHR    9-61#    9-62    44-10    44-16    45-9     45-61    62-67*   84-10    86-57    93-6     100-9    112-52   113-22   115-11
        115-26   131-72*  132-18   132-39*  139-24   140-29   141-39   143-50   143-64   143-68   143-125  143-186
S.SDCL    9-57#    9-58    45-47    45-51    131-109* 131-113*
S.SEEK    9-56#    9-57    120-57   120-66*  143-41*
S.TGOF    9-71#    9-72    83-5     91-11    91-13    92-8     92-9     140-21   140-23*  141-6    141-59   141-60
S.TGSS    9-72#   62-67    62-67    83-15    87-27    88-22    91-15    92-14    92-21    130-6    140-13   141-64   141-70
S.TRKL    9-60#    9-61    44-13    77-14    77-16    84-5     86-48    89-13    90-16    94-10    131-92*
SBCRES    8-46#   62-67
SCHAR0   62-62   126-54   131-19#  131-154  138-19   138-97
SCHAR1   62-63   131-115  138-19#  143-19
SCHRCP  132-20   132-30#
SCR      62-67    62-67#
SCR1     45-14*   46-16#   50-136*  63-77*   63-80*   75-35*   75-87*   77-10*   87-8*    88-6*    89-6*    90-6*    97-59    145-42*
```

CROSS REFERENCE TABLE (CREF V04.00)

```
        145-44*  145-66*  145-69*
SCR2     46-17#  102-66*  105-110* 106-93*  106-105* 106-112  106-114* 107-41*  107-88*  109-53*  110-99*  112-79
SCTWRD    9-76#   99-91   110-75   110-84
SDIS     41-5#    41-6     41-6     41-6#    41-8     41-8     41-8#    41-9     41-9     41-9#    41-10    41-10    41-10#   41-11
         41-11    41-11#   41-12    41-12    41-12#   41-13    41-13    41-13#   41-15    41-15    41-15#   41-16    62-67    62-67
         62-67    62-67    62-67    62-67    62-67    62-67#   62-67#   62-67#   62-67#
SDIVER    7-4#
SEARCH  112-83   114-2#
SEC512    5-72#
SEC576    5-73#
SECCHK   62-41   102-70   105-19#  106-19   111-54
SECCYL   62-67#  134-17   136-20*  137-13
SECGRP   62-67#  136-13*  136-17   137-20   142-13
SECMAX   46-38#   62-67*   63-99    63-101
SECPTR   96-15*   97-14    98-11#
SECTOR   46-11#   97-40
SECTRK   62-67#  131-91*  136-10   142-15
SEEK     56-24    62-49    63-114   112-70   113-33   115-47   117-19#  120-19   120-87   121-76
SEKCNT  120-59   120-66#
SEKEXT  117-67   117-74   117-85   117-89#
SEKINP   10-19#  117-44   117-63   119-30   120-52   120-85
SEKOUT  117-53   117-88#
SEKREQ   10-35#   50-44    56-22    60-35    60-44   118-19   120-55   121-69
SEKTST   62-50   117-83   120-19#  120-44   120-99   121-19
SEND      6-7#    40-16    55-49    62-67
SEQBE    62-34    64-44    75-19#   82-19
SEQLP    49-11#   56-53
SEQNCR   48-36    49-2#    62-67
SEQSEK   10-51#   50-85    62-67    63-70    64-34   145-40   145-64
SEQTG    62-36    64-46    86-19#   95-19
SER0     57-19    62-84#
SER1     57-56    62-84#
SER10    62-67    62-84#
SER11    62-67    62-84#
SER12    62-67    62-84#
SER13    62-67    62-84#
SER14    62-67    62-84#
SER15    62-67    62-84#
SER16    62-67    62-84#
SER17    62-67    62-84#
SER18    62-67    62-84#
SER18A   62-67    62-84#   122-64
SER18B   62-67    62-84#   122-65
SER18C   62-67    62-84#   122-66
SER18D   62-67    62-84#   122-67
SER18E   62-67    62-67#
SER18F  122-54   122-64#
SER19    62-84#  101-169  105-147
SER2     57-13    62-84#
SER20    62-84#  134-22   135-21   135-23   135-30   135-53   135-67   135-69   139-18   139-36   139-42   140-45   141-29   145-71
SER21    62-84#  106-73   106-107  107-60   107-92
SER22    40-73    40-77    40-83    53-29    57-35    62-84#  100-16   100-45   101-167  101-175  103-12   105-145  105-153  120-94
        120-110
SER24    62-84#  110-98
SER25    62-84#  110-101
SER26    62-84#  113-14   114-48   115-38
```

CROSS REFERENCE TABLE (CREF V04.00)

```
SER3     57-64     62-84#
SER32    62-84#    113-16    114-50    115-40
SER36    62-84#    101-166   105-144
SER39    51-23      62-67     62-84#
SER4     62-84#    109-70
SER40    62-67      62-84#
SER41    62-67      62-84#
SER42    57-28      62-84#
SER43    57-32      62-84#
SER44    57-34      62-84#
SER45    62-67      62-84#
SER5     62-84#    131-150
SER6     62-84#    125-38
SER7     62-84#    121-65
SER8     62-84#    124-37
SER9     62-84#    119-33
SERCHN   143-115*  143-124*  143-134*  143-135*  143-136*  143-137*  143-139   143-158*  143-168   144-4#
SERRTY   143-61*   143-191*  144-3#
SERSEC   143-177   143-179*  143-196*  143-197*  143-198*  143-199*  143-201*  143-207   143-216   143-217   143-218   143-219   144-14    144-22#
SETSEC    89-9      89-25     93-2#     94-6
SETUP    56-49      62-31     63-19#    64-19     70-9      99-58     101-86    101-94    101-140   102-61    105-107   111-69    116-20    145-86
SHRTTO    7-3#     124-63
SLOP     62-67      62-67#
SMASKL   131-46#   131-49
SMASKX   131-46    131-50#
SNDAGN    37-14#    37-18
SNDEX    109-71    109-85#
SNDONE    40-10     41-7#
SORT     62-61     125-36    126-19#   131-19
SORTBB   126-47    129-2#
SORTBE   126-46    127-2#
SORTTG   126-44    130-2#
SPINUP    62-60    124-81    125-19#   126-19
SS         7-9#
ST        40-42     41-38#    55-56     55-71     55-80     55-84*    57-22     57-41     57-45     62-67     62-67     124-42    124-43    124-44
         124-45    124-52    124-55    124-59    124-63    124-76    131-60    131-83    131-89    131-107   131-108   131-110   132-31    133-37
         133-40    135-38    135-50*   135-52*   135-53    135-53    136-8     136-15    137-10    137-18
ST.C       6-37#
ST.CON     6-36#
ST.DB      6-57#
ST.DE      6-51#     57-48
ST.DF      6-54#
ST.DR      6-46#
ST.EL      6-48#     55-84
ST.ERR     6-35#     55-80     57-45
ST.FO      6-56#
ST.MOD     6-34#     57-41
ST.MSK     6-32#
ST.OA      6-44#
ST.PE      6-53#
ST.PS      6-49#     57-23
ST.RE      6-52#
ST.REQ     6-33#     55-84*    57-22
ST.RR      6-45#     57-38
ST.RTY     6-38#
ST.RU      6-50#     57-23     57-26
```

CROSS REFERENCE TABLE (CREF V04.00)

```
ST.S7    6-58#
ST.SR    6-47#   57-23   57-30
ST.UNT   6-31#
ST.WE    6-55#
STACK   34-19    34-32#
STACYL  45-36*   45-39*  45-48*  45-52*  46-4#   143-51*  143-52*
START   34-27    62-67#
STATUS   6-10#   35-32
STOP    48-48    48-52#  51-27   62-67   62-67
STSERR 120-49   120-82#
STSEXT 120-46   120-81  120-88  120-95  120-103  120-107  120-112#
STSRES   8-47#   41-8
SUBTRV 132-3    132-13#
SUBUNT  62-67#  131-51*
SUNIT  143-33*  143-100 143-208 143-212 143-223* 143-224* 144-2#
SUSETL  62-67    62-67#
SWAPBE 127-23   128-2#
T1MSIZ   8-4#
T2CMD    8-6#
T2DLL    8-5#
T4BB1    8-9#    62-67   62-67
T4BB2    8-10#   62-67
T4MPRM   8-7#    62-67
T4MXFR   8-13#   54-1    99-125  111-66
T4SEEK   8-12#  120-62
T4SOFT   8-11#   36-7    40-106   56-13   100-51   101-144  103-30   105-122  106-92   107-85   117-68   120-70   120-91   143-101
T4UPRM   8-8#    62-67   62-67   143-220
TALK    40-2#    48-41   57-10    57-16    57-53    57-63    109-67   119-10   121-40   124-34   125-33   131-54   143-74   143-81
TG$    140-33*  140-36*  140-45  141-29  141-91#
TGS     62-67#  140-12  140-38  141-5    141-45
THREBE  73-48    73-51#
TLEN.U   9-78#    9-79   62-67
TLKHST  36-2#    40-50   95-32
TMEMPL  95-42*   98-6*   98-7    98-10#
TO     124-61   124-65  124-86#
TRACK   46-10#   97-6    143-135
TRACKS  10-53#   62-67    84-8    86-37    86-43    86-54    140-30   141-40   142-11
TRAV   131-65   132-2#  132-9
TRK$    62-84#  140-36
TRKGRP   7-27#   84-11   86-58   93-7    136-8    140-35   141-44
TROOT   62-67    62-67#
TRYAGN  73-44#   73-52
TSTNEC 117-69   118-2#
U.CBN    9-41#    9-42    63-57   63-59*  63-61   63-63*  63-85*  63-87*  70-2*   70-4*   95-40   95-41   99-50   99-50
        101-46   101-47  101-101 101-102* 101-104 101-106* 102-53  102-53  104-7   104-8   105-67  105-68* 105-70  105-72*
        108-46   108-47* 108-49  108-51*  109-48  109-48  112-61* 112-66* 113-17  113-17  113-24  113-25* 113-27  113-29*
        115-7    115-8   115-20* 115-21*  115-41  115-41  116-13* 116-17* 117-59  117-59  118-13  118-14  120-71  120-71
U.CCNT   9-20#    9-21    87-12   87-14*   87-16   87-23*  88-10   88-12*  88-14   88-21*  91-8*   91-10*  92-11*  92-13*
U.CCOP   9-44#    9-45   112-56*  113-6   113-21*  115-9   143-63* 143-181 143-183*
U.CCYL   9-48#    9-49    95-46   95-48   100-15   100-15  101-133 101-133 101-173 101-173 103-11  103-11  105-100 105-100
        105-151  105-151 106-77  106-78  106-111  106-112 107-64  107-65  107-89  107-89  107-96  107-97  109-50  109-50
        110-105  110-105 118-21  119-13  120-72   120-73  120-92  120-93  120-109 120-109 121-72* 121-73* 123-36* 123-37*
U.CGRP   9-49#    9-50    95-50   100-15  101-133  101-173 103-11  105-100 105-151 106-77  106-111 107-64  107-89  107-96
        109-49   110-105 120-72  120-92  120-109  121-74* 123-38*
U.COPY   9-43#    9-44   113-8   124-80*  143-59  143-184
U.CSEC   9-26#    9-27    50-20*  50-68*   63-43   63-56*  63-89*  63-91   70-6*   101-98  101-99* 105-61  105-62* 108-36
```

CROSS REFERENCE TABLE (CREF V04.00)

```
        108-37*  112-72   116-11
U.CTRK    9-22#    9-23    89-14    89-17*   90-11    90-14*   93-9*
U.ECCT    9-32#    9-33   107-55   124-58*  143-172
U.ELEV    9-29#    9-30    63-108* 106-73   106-89   106-107  107-60   107-76   107-92   109-42   109-45   109-78   109-80*
U.LCYL    9-50#    9-51   101-134  101-134  101-174  101-174  105-101  105-101  105-152  105-152  119-17   120-92   120-92   120-108
        120-108
U.LGRP    9-51#    9-78    62-67   101-134  101-174  105-101  105-152  120-92   120-108
U.MASK    9-27#    9-28    35-16    40-8     40-44    55-45    60-55    62-67*   100-8    103-7    143-138
U.MBN     9-40#    9-41    63-84    63-86    70-1     70-3     74-64    75-43    75-79    75-81*   75-83    75-85*   75-91    75-107
         76-9     81-12    81-14    81-56    81-60    84-29*   84-32*   87-31    87-32*   87-34    87-36*   88-30    88-32*   88-35
         88-37*   89-18    89-19*   89-21    89-23*   89-31    89-33*   89-35    89-37*   90-15    90-17*   90-19    90-21*   91-12*
         91-14*   92-31*   92-32*   94-12    94-13*   94-15    94-17*   112-73   112-75   116-12   116-14
U.MLEV    9-31#    9-32    63-107   106-90   107-78   124-54*
U.MSEC    9-24#    9-25    63-106*  95-34   101-178*  105-156* 108-45*  112-58*
U.MSTO    9-16#    9-17    55-12    117-78   124-75*  143-84
U.NEXT    9-10#    9-11    48-27    62-67    62-67    62-67    62-67
U.NFUN    9-18#    9-19    49-10    56-54*   62-67*
U.NSEC    9-23#    9-24    50-21*   50-69*   63-44    63-58    63-112*  70-7*    99-57*   101-78*  101-91   101-93*  101-97   101-100
        101-108* 101-139* 102-60*   105-60   105-63   105-66   105-74*  105-106* 108-35   108-38   108-41   108-43*  111-34   111-36*
        114-30*  116-10*
U.PARM    9-37#    9-38    42-8     45-21    48-28    48-37    50-10    50-15*   50-66*   50-108   50-110*  50-131   50-133*  56-46
         56-48*   58-22    58-24*   59-28    59-36*   62-67    62-67*   62-67*   63-65    63-75*   63-83*   63-92    65-12    65-15*
         66-16*   68-18*   69-18*   75-32    75-140   78-13    78-15*   79-13    79-15*   81-26    81-46    81-53    86-32    87-19
         87-21*   88-17    88-19*   95-33    96-6     97-79    99-51    99-53    99-55*   101-83   101-85*  101-119  101-137* 101-171
        102-50   102-54   102-56   102-58*  105-85   105-104* 105-149  106-57   106-75   106-80   106-84   106-109  107-62   107-94
        107-103  107-112  109-49   109-52   110-103  111-57   111-59*  112-37   112-47*  114-24   114-26*  116-6    116-8*   117-43
        117-60   117-62   117-64*  118-7    119-29   119-31*  120-51   120-53*  120-72   120-84   120-86*  122-58   122-60*  123-32
        123-34*  145-50   145-52*
U.PAT     9-19#    9-20    67-15*   99-68    99-88    99-110
U.PCTG    9-21#    9-22    87-24    87-30*   88-24    88-33*   91-17*   92-29*
U.RBN     9-42#    9-43    95-37    95-38    99-51    99-51    101-131  101-132  101-171  101-172  102-54   102-54   105-98   105-99
        105-149  105-150  106-76   106-76   106-110  106-110  107-63   107-63   107-95   107-95   109-49   109-49   110-104  110-104
        112-54*  112-55*  114-18   114-19*  114-21   114-23*  114-37   114-38*  114-40   114-42*  117-60   117-60   118-10   118-11
        120-72   120-72
U.RCOV    9-38#    9-39    50-45*   51-43    53-10    55-27    55-31*   56-8     56-18    56-21*   60-7     60-9*    60-16    60-18*
         60-27*   60-36*   60-45*   60-59*   62-67*   63-120*  108-62   108-64*  109-39   109-72   109-77*  117-48   118-18   120-54
        120-56*  120-76   121-67   121-70*
U.RRTY    9-15#    9-16    63-113*  106-73   106-87   106-107  107-60   107-74   107-92   108-33   108-55*  109-83*
U.RTRY    9-30#    9-31    108-52   124-51*
U.RVER    9-46#    9-47    112-51*  113-11   114-45   115-35
U.RWER    9-45#    9-46    100-30*  100-44   101-81   101-109  103-24*  105-75   112-50
U.RWTO    9-17#    9-18    63-110*  99-40    99-42*   100-49   101-170  102-40   102-42*  103-28   105-148  108-57*  112-69*  113-32*
        115-46*
U.SDI2    9-35#    9-36    41-6     41-8     41-9     41-10    41-11    41-12    41-13    41-15    50-48    62-67    62-67    62-67
U.SDIL    9-34#    9-35    40-30    124-62*
U.SDIS    9-33#    9-34    40-27    55-55    62-67*   124-66*
U.SRTY    9-14#    9-15    117-47*  117-54   117-75   117-77*  120-68
U.SUBP    9-11#    9-12    50-18    50-41    50-57    50-92    58-6     59-6     62-67    62-67    122-41   126-35   131-122  131-135
        132-15   138-69   138-82   143-36   145-34   145-55
U.SUBU    9-39#    9-40    36-7     40-106   50-38*   50-58    50-77    50-78    50-94    50-112*  50-115   54-4     56-2     56-13
         58-18*   59-24*   100-37   100-51   101-144  103-30   105-122  106-92   107-83   107-89   117-68   120-61   120-70   120-91
        131-35   131-118  131-124  131-130  131-132* 131-145* 138-47   138-65   138-71   138-77   138-79*  138-92*
U.TIMH    9-12#    9-13    117-79*  120-104  120-106* 143-85*  143-107  143-109*
U.TIML    9-13#    9-14    117-81*  120-100  120-102* 143-87*  143-103  143-105*
U.TSEC    9-25#    9-26    50-22*   50-70*   63-45    63-78    63-90    74-62*   75-77    75-109   77-17*   81-65*   85-16*   86-64*
U.UNUM    9-47#    9-48    36-7     40-106   50-77    54-3     56-1     56-13    62-67    62-67*   100-51   101-144  103-30   105-122
```

CROSS REFERENCE TABLE (CREF V04.00)

	106-92	107-82	107-89	117-68	120-60	120-70	120-91	122-46	131-117	138-47	138-64	143-33	143-34	
U.WPRT	9-36#	9-37	57-59	62-67*	100-36	145-85*								
U.WRIT	9-28#	9-29	57-44*	57-58	100-32									
UCURSR	62-67	62-67	62-67	62-67	62-67	62-67#	62-67*	62-67*	62-67*					
UDA50	2-40#	2-48	2-63	2-75	5-10	5-41	5-97	7-54	9-81	34-5	42-17	50-142	61-6	62-67
	62-67	62-67	62-67	63-3	63-17	63-125	64-3	64-17	72-12	72-18	73-3	73-17	74-26	74-32
	74-74	75-3	75-17	81-87	82-3	82-17	82-43	82-49	83-12	84-25	85-15	85-23	86-3	86-17
	92-18	94-20	95-3	95-17	96-22	96-45	97-26	98-13	99-3	99-17	99-45	100-55	101-3	101-17
	101-54	101-90	101-184	102-3	102-17	102-45	104-27	105-3	105-17	105-45	105-171	106-3	106-17	106-43
	106-61	106-136	107-3	107-17	107-126	108-3	108-17	108-70	109-3	109-17	109-87	110-3	110-17	110-129
	111-3	111-17	111-39	111-74	112-3	112-17	115-28	116-23	117-3	117-17	119-36	120-3	120-17	120-114
	121-3	121-17	121-80	122-3	122-17	122-70	123-3	123-17	123-44	124-3	124-17	124-100	125-3	125-17
	125-46	126-3	126-17	130-27	131-3	131-17	131-134	135-29	135-40	137-33	138-3	138-17	138-81	141-76
	141-79	142-20	143-3	143-17	143-113	143-156	144-20	145-3	145-17	145-92				
UDA52	2-41#	2-54	2-63	2-80	5-16	5-46	5-102	7-59	9-86	34-10	42-17	62-67	63-22	64-22
	72-12	73-22	74-26	74-32	75-22	82-22	82-43	82-49	83-12	84-25	85-15	86-22	92-18	95-22
	96-27	96-50	97-31	97-64	99-22	99-45	100-22	101-22	101-59	101-90	102-22	102-45	103-17	105-22
	105-50	106-22	106-49	106-61	107-22	107-43	108-22	109-22	110-22	111-22	111-44	112-22	115-28	117-22
	120-22	121-22	122-22	122-85	123-22	124-22	125-22	126-22	131-22	131-134	135-29	135-40	138-22	138-81
	141-76	141-79	143-22	143-118	143-144	143-161	144-7	145-22	145-106					
UDADM4	2-77#													
ULOP	62-67	62-67#												
UMASK	62-67	62-67#	62-67*	62-67*										
UNADDR	7-56#	62-67	96-35	101-71	104-18	111-52								
UNITS	62-67	62-67	62-67	62-67	62-67	62-67	62-67	62-67	62-67#					
UNSSUC	8-44#	40-85												
UP	75-34	75-87#												
UPG	86-47	89-2#												
UPT	86-45	87-2#	89-39											
UREAD	6-14#	51-16												
UTOTST	8-14#	62-67												
UWRITE	6-15#													
VALBIT	62-67	62-67#												
VALID	6-39#	40-84	40-87	55-9	55-37									
WAITSI	6-13#	100-12	103-8	143-140										
WBLOCK	99-115	100-2#												
WBUFLN	5-94#	5-95	62-67	96-13										
WCHECK	10-54#	50-14	62-67	63-66	63-82	65-14	68-8	68-17	69-10					
WCHKAL	10-56#	62-67	68-12											
WCONT	5-69#													
WONLY	10-47#	62-67	66-8	138-38										
WREAL	5-70#	97-11												
WRITBT	41-23#	57-61*												
WRITE	62-38	97-84	99-19#	101-19										
WSTOP	5-71#	97-13												
XBNCYL	7-40#	131-108												
XFERRT	7-5#													
XOPLP0	110-67#	110-79												
XOPLP1	110-71#	110-78												
XOPLP2	110-81#	110-85												
XREAD	6-5#	103-23	143-150											
XWRITE	6-6#	100-28												
ZEREDC	39-13	39-15#												

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
        107-89  107-89  107-89  107-89  107-89  107-89  107-92  107-92  107-92  107-92  107-93  107-93  107-93  107-94
        107-94  107-95  107-95  107-95  107-96  107-96  107-96  107-97  107-97  107-97  109-48  109-48  109-48  109-48
        109-49  109-49  109-49  109-49  109-49  109-50  109-50  109-70  110-97  110-98  110-101 110-102 110-102 110-102
        110-103 110-103 110-104 110-104 110-104 110-105 110-105 110-105 110-105 110-122 110-122 110-122 110-122 110-122
        110-122 110-122 110-122 110-122 110-122 110-122 110-123 110-123 110-123 110-123 113-10  113-14  113-16  113-17
        113-17  113-17  113-17  114-44  114-48  114-50  114-51  114-51  115-34  115-38  115-40  115-41  115-41  115-41
        115-41  117-59  117-59  117-59  117-59  117-60  117-60  117-60  117-60  117-60  117-60  117-60  119-33  120-71
        120-71  120-71  120-71  120-71  120-72  120-72  120-72  120-72  120-72  120-72  120-73  120-92  120-92  120-92
        120-92  120-92  120-92  120-93  120-94  120-108 120-108 120-108 120-108 120-109 120-109 120-109 120-110 121-65
        124-37  125-38  131-150 134-22  134-22  135-21  135-21  135-23  135-23  135-30  135-30  135-47  135-48  135-53
        135-53  135-53  135-53  135-65  135-66  135-67  135-67  135-69  135-69  135-69  135-69  138-40  138-47  139-18
        139-18  139-36  139-36  139-36  139-36  139-42  139-42  139-42  139-42  140-45  140-45  140-45  140-45  141-29
        141-29  141-29  143-207 143-207 145-71  145-71
MSG      12-20#  41-6    41-8    41-9    41-10   41-11   41-12   41-13   41-15   62-67   62-67   62-67
MSSG     15-138# 50-77   107-89  138-47
NDERR    15-124# 35-12   40-73   40-77   53-21   53-29   57-35   99-52   100-16  100-45  101-135 101-167 101-175 102-55
        103-12  105-102 105-145 105-153 106-79  106-113 107-66  107-98  109-51  110-124 113-18  114-52  117-61  120-74
        120-94  120-110 145-72
ONETIM   16-31#  62-67
OVTERM    2-77    2-77#   2-77#   62-84   62-84#  63-5    63-5#   64-5    64-5#   73-5    73-5#   75-5    75-5#   82-5
         82-5#   86-5    86-5#   95-5    95-5#   99-5    99-5#   101-5   101-5#  102-5   102-5#  105-5   105-5#  106-5
        106-5#  107-5   107-5#  108-5   108-5#  109-5   109-5#  110-5   110-5#  111-5   111-5#  112-5   112-5#  117-5
        117-5#  120-5   120-5#  121-5   121-5#  122-5   122-5#  123-5   123-5#  124-5   124-5#  125-5   125-5#  126-5
        126-5#  131-5   131-5#  138-5   138-5#  143-5   143-5#  145-5   145-5#  145-113
POP      14-11#  35-42   36-7    37-23   39-17   40-21   40-26   40-51   40-89   40-100  40-106  40-107  42-25   50-77
         50-137  51-19   51-28   51-38   55-63   55-70   56-13   56-17   56-31   56-51   60-63   62-67   62-67   62-67
         62-67   62-67   62-67   62-67   62-67   62-67   62-67   62-67   62-67   62-67   62-67   62-67   62-67   62-67
         71-26   72-15   74-34   74-41   74-63   75-117  80-13   81-38   81-42   81-61   83-22   84-26   92-30   97-23
         99-109  100-18  100-29  100-51  101-77  101-144 103-30  104-24  105-122 106-92  107-52  107-58  107-68  107-87
        107-89  110-87  110-90  110-94  117-68  119-23  120-70  120-91  122-61  126-48  126-51  128-23  129-32  131-50
        131-55  131-66  131-73  131-156 132-4   132-21  132-38  132-41  133-50  135-72  137-30  138-47  138-99  139-45
        141-84  141-88  143-222 143-226
PUSH     14-3#   35-30   36-7    37-12   39-7    40-7    40-22   40-31   40-54   40-106  42-7    50-9    50-77   51-6
         51-20   51-32   52-2    53-6    55-11   56-13   60-52   62-67   62-67   62-67   62-67   62-67   62-67   62-67
         62-67   62-67   62-67   62-67   62-67   62-67   62-67   71-7    72-7    74-5    74-15   74-55   75-106  80-7
         81-8    81-28   83-4    84-4    84-18   92-7    97-19   99-87   100-6   100-51  101-45  101-144 103-30  104-6
        105-122 106-92  107-40  107-81  107-89  110-39  110-91  117-68  119-16  120-70  120-91  122-36  126-33  126-39
        128-7   129-19  131-33  131-42  131-53  131-64  131-70  132-2   132-19  132-30  133-7   135-8   137-7   138-33
        138-47  139-7   140-11  141-61  143-32  143-42
REPSFT   15-9#   36-7    40-106  56-13   100-51  101-144 103-30  105-122 106-92  117-68  120-70  120-91
RXOR     16-10#  39-11
SOFT     15-4#   40-24   40-40   40-56   40-60   40-64   40-68   40-72   40-82   53-28   101-114 101-145 101-150 101-154
        101-159 101-164 105-80  105-114 105-118 105-123 105-128 105-132 105-137 105-142 106-73  106-107 107-53  107-59
        107-92  120-71  120-92  120-108
SYSFTL   15-65#
TEXT     31-145#  62-84
```